UNITED STATES AIR FORCE

SUMMER RESEARCH PROGRAM -- 1998

SUMMER RESEARCH EXTENSION PROGRAM FINAL REPORTS

VOLUME 5

ARNOLD ENGINEERING CENTER
AIR LOGISTICS CENTERS
UNITED STATES AIR FORCE ACADEMY

RESEARCH & DEVELOPMENT LABORATORIES

5800 Uplander Way

Culver City, CA 90230-6608

Program Director, RDL
Gary Moore

Program Manager, AFOSR
Colonel Cerveny

Program Manager, RDL
Scott Licoscos

Program Administrator, RDL
Johnetta Thompson

Program Administrator,RDL
Rebecca Kelly-Clemmons

Submitted to:

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Bolling Air Force Base

Washington, D.C.

December 1998

PREFACE

This volume is part of a four-volume set that summarizes the research of participants in the 1998 AFOSR Summer Research Extension Program (SREP). The current volume, Volume 1 of 5, presents the final reports of SREP participants at Armstrong Laboratory.

Reports presented in this volume are arranged alphabetically by author and are numbered consecutively -- e.g., 1-1, 1-2, 1-3; 2-1, 2-2, 2-3, with each series of reports preceded by a 35 page management summary. Reports in the five-volume set are organized as follows:

VOLUME	TITLE
1	Armstrong Research Laboratory
2	Phillips Research Laboratory
3	Rome Research Laboratory
4	Wright Research Laboratory
5	Air Logistics Center Arnold Engineering Development Center

REPORT DOCUMENTATION PAGE

Form Approved

AFRL-SR-BL-TR-00-

0778

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT
	December, 1998	

4. TITLE AND SUBTITLE
1998 Summer Research Program (SRP), Summer Research Extension Program (SREP), Final Report, Volume 5, Arnold Eng. Development Center (AEDC) and Air Logistics Centers

5. FUNDING NUMBERS
F49620-93-C-0063

6. AUTHOR(S)
Gary Moore

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Research & Development Laboratories (RDL)
5800 Uplander Way
Culver City, CA 90230-6608

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research (AFOSR)
801 N. Randolph St.
Arlington, VA 22203-1977

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The United States Air Force Summer Research Program (SRP) is designed to introduce university, college, and technical institute faculty members to Air Force research. This is accomplished by the faculty members, graduate students, and high school students being selected on a nationally advertised competitive basis during the summer intersession period to perform research at Air Force Research Laboratory (AFRL) Technical Directorates and Air Force Air Logistics Centers (ALC). AFOSR also offers its research associates (faculty only) an opportunity, under the Summer Research Extension Program (SREP), to continue their AFOSR-sponsored research at their home institutions through the award of research grants. This volume consists of a listing of the participants for the SREP and the technical report from each participant working at the Arnold Engineering Development Center (AEDC) and Air Logistics Centers.

14. SUBJECT TERMS
Air Force Research, Air Force, Engineering, Laboratories, Reports, Summer, Universities, Faculty, Graduate Student, High School Student

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Standard Form 298 (Rev. 2-89) (EG)
Prescribed by ANSI Std. 239.18
Designed using Perform Pro, WHS/DIOR, Oct 94

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	**PR** - Project
G - Grant	**TA** - Task
PE - Program Element	**WU** - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with....; Trans. of....; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
	Leave blank.
NASA -	Leave blank.
NTIS -	

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self- explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

1998 SREP Final Technical Report Table of Contents

Armstrong Research Laboratory

Volume 1

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
1	Dr. Gerald P. Chubb	Scoring Pilot Performance of Basic Flight Maneuvers Ohio University	AFRL/HEA
2	Dr. Brent D. Foy	Development & Validation of a Physiologically-Based Kinetic Model of Perfused Liver for water-soluble Compounds Wright State University	AFRL/HES
3	Dr. Charles Lance	Extension of Job Performance Measurement Technologies to Development of a Prototype Methodology for Assessing Work Team University of Georgia Research Foundation	AFRL/HEJ
4	Dr. David Woehr	Validation of the Multidimensional Work Ethic Profile (MWEP) as a Screening Tool for AF Enlisted Personnel Texas A & M University College Station	AFRL/HEJ

1998 SREP Final Technical Report Table of Contents

Phillips Research Laboratory

Volume 2

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
1	Dr. Mark J. Balas	Non-Linear Adaptive Control for a Precision Deployable Structure with White ligh University of Colorado at Boulder	AFRL/VSDD
2	Dr. Neb Duric	Image Recovery Using Phase Diversity University of New Mexico	AFRL/DEBS
3	Dr. George W. Hanson	Perturbation Analysis of the Natural Frequencies Targets in Inhomogeneous University of Wisconsin-Milwaukee	AFRL/DEHP
4	Dr. Brian D. Jeffs	Bayesian restoration of Space object Images from Adaptive Optics Data with unknown data Brigham Young University	AFRL/DES
5	Dr. Aravinda Kar	Effects of Vapor-Plasma Layer on Thick-Section Cutting and Calculation of Modes University of Central Florida	AFRL/DEOB
6	Dr. Donald J. Leo	Adaptive Vibration suppression for autonomous Control Systems University of Toledo	AFRL/VSDV
7	Dr. Hanli Liu	Continuous- Wave approach to 3-D imaging Through Turbid media w/a Single Planar Measurement University of Texas Arlington	AFRL/DEBS
8	Dr. Joshua C. Biefang	Optical Clocks Based on Diode Lasers University of New Mexico	AFRL/ DELO
9	Dr. Eric J. Paulson	Optimization 7 Analysis of a Waverider Vehicle For Global Spaceplane University of Colorado at Boulder	AFRL/PRR
10	Dr. Kenneth F. Stephens II	Simulation of an explosively Formed Fuse Using MACH 2 University of North Texas	AFRL/DEHE

1998 SREP Final Technical Report Table of Contents

Rome Research Laboratory

Volume 3

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
1	Dr. Milica Barjaktarovic	Specification and Verification of SDN. 701 MSP Functions and Missi Crypto Wilkes University	AFRL/IFGB
2	Dr. Stella N. Batalama	Robust Spread Spectrum Communications: Adaptive Interference Mitigation SUNY Buffalo University	AFRL/IFGC
3	Dr. Nikolaos G. Bourbakis	Hierarchical-Adaptive Image Segmentation SUNY Binghamton University	AFRL/IRE
4	Dr. Venugopala R. Dasigi	Information Fusion w/Multiple Feature Extractors for automatic Text Sacred Heart University	AFRL/IRE
5	Dr. Richard R. Eckert	The Interactive Learning Wall; A PC-Based, Deployable Data Wall for Use in a College Classroom SUNY Binghamton University	AFRL/IFSA
6	Dr. Kuo-Chi Lin	Web-Based Distributed Simulation University of Central Florida	AFRL/IFSB
7	Dr. Dimitrios N. Pados	Adaptive Array Radars and Joint Space-Time Auxiliary Verctor Filtering	AFRL/SN
8	Dr. Brajendra N. Panda	Information Warfare" Design of an Efficient Log Management Method to Aid In Data University of North Dakota	AFRL/IFGB
9	Dr. Michael A Pittarelli	Complexity of Detecting and content-driven methods for resolving database SUNY of Tech Utica	AFRL/IFTB
10	Dr. Mark S. Schmalz	Errors Inherent in 3D Target Reconstruction from Multiple Airborne Images University of Florida	AFRL/IRE
11	Dr. Nong Ye	Model-based Assessment of Campaign Plan-Performance under Uncertainty Arizona State University	AFRL/IFSA
12	Mr. Parker Bradley	Development of User-Friendly CompEnvironment for Blind Source Separation Syracuse University	AFRL/IFGC

1998 SREP Final Technical Report Table of Contents

Wright Research Laboratory

Volume 4

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
1	Dr. Brian P. Beecken	Development of a statistical Model predicting the impact of a scent Projector's Nonuniformity on a test Article's Image Bethel College	AFRL/MN
2	Dr. John H. Beggs	Implementation of an Optimization Algorithm in Electromagnetics for Radar absorbing Material Layers Mississippi State University	AFRL/VASD
3	Dr. Raj Bhatnagar	Analysis of Intra-Class Variability and synthetic Target Models Models for Use in ATR University of Cincinnati	AFRL/SN
4	Dr. Gregory Blaisdell	Validation of a Large Eddy Simulation Code & Development of Commuting Filters Purdue University	AFRL/VAAC
5	Dr. John Douglas	Roles of Matched Filtering and Coarse in Insect Visual Processing University of Arizona	AFRL/MN
6	Dr. William Hosford	Prediction of Compression Textures in Tantalum Using a Pencil-Glide Computer Mode Program University of Michigan	AFRL/MN
7	Dr. Yi Pan	Parallelization of Time-Dependent Maxwell Equations Using High Perform University of Dayton	AFRL/VASD
8	Dr. Kishore Pochiraju	A Hybrid Variational-Asymptotic Method for the Analysis of MicroMechanical Damage in Composites Stevens Institute of Technology	AFRL/MLBM
9	Dr. Yuri Shtessel	Continuous Sliding Mode Control Approach for Addressing Actuator Deflection and Deflection rate Saturaton in Tailess Aircraft Control and Re-Configurable Flight Control University of Alabama In Huntsville	AFRL/VACD
10	Dr. Janusz Starzyk	Feature Selection for Automatic Target Recognition: Mutual Information & Statistical Techniques Ohio University	AFRL/SN

1998 SREP Final Technical Report Table of Contents

Volume 5

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
		Arnold Engineering Development Center	
1	Dr. Frank Collins	Monte Carlo Compution of Species Separation by a Conical Skimmer in Hypersonic Transition Flow University of Tennessee Space Institute	AEDC
		Air Logistics Centers	
2	Dr. Paul W. Whaley	Probabilistic Analysis of Residual Strength in Corroded and Uncorroded Aging Air Mineralization Oklahoma Christian University of Science & Art	OCALC/TIE
3	Dr. Devendra Kumar	Further Development of a Simpler, Multiversion Control Protocol for Internet Databases University of Georgia	SAALC
4	Dr. Joe G. Chow	An Automated 3-D Surface Model Creation Module for Laser Scanned Point Data Florida International University	WRALC

1998 SUMMER RESEARCH EXTENSION PROGRAM (SREP) MANAGEMENT REPORT

1.0 BACKGROUND

Under the provisions of Air Force Office of Scientific Research (AFOSR) contract F49620-90-C-0076, September 1990, Research & Development Laboratories (RDL), an 8(a) contractor in Culver City, CA, manages AFOSR's Summer Research Program. This report is issued in partial fulfillment of that contract (CLIN 0003AC).

The Summer Research Extension Program (SREP) is one of four programs AFOSR manages under the Summer Research Program. The Summer Faculty Research Program (SFRP) and the Graduate Student Research Program (GSRP) place college-level research associates in Air Force research laboratories around the United States for 8 to 12 weeks of research with Air Force scientists. The High School Apprenticeship Program (HSAP) is the fourth element of the Summer Research Program, allowing promising mathematics and science students to spend two months of their summer vacations working at Air Force laboratories within commuting distance from their homes.

SFRP associates and exceptional GSRP associates are encouraged, at the end of their summer tours, to write proposals to extend their summer research during the following calendar year at their home institutions. AFOSR provides funds adequate to pay for SREP subcontracts. In addition, AFOSR has traditionally provided further funding, when available, to pay for additional SREP proposals, including those submitted by associates from Historically Black Colleges and Universities (HBCUs) and Minority Institutions (MIs). Finally, laboratories may transfer internal funds to AFOSR to fund additional SREPs. Ultimately the laboratories inform RDL of their SREP choices, RDL gets AFOSR approval, and RDL forwards a subcontract to the institution where the SREP associate is employed. The subcontract (see Appendix 1 for a sample) cites the SREP associate as the principal investigator and requires submission of a report at the end of the subcontract period.

Institutions are encouraged to share costs of the SREP research, and many do so. The most common cost-sharing arrangement is reduction in the overhead, fringes, or administrative charges institutions would normally add on to the principal investigator's or research associate's labor. Some institutions also provide other support (e.g., computer run time, administrative assistance, facilities and equipment or research assistants) at reduced or no cost.

When RDL receives the signed subcontract, we fund the effort initially by providing 90% of the subcontract amount to the institution (normally $18,000 for a $20,000 SREP). When we receive the end-of-research report, we evaluate it administratively and send a copy to the laboratory for a technical evaluation. When the laboratory notifies us the SREP report is acceptable, we release the remaining funds to the institution.

2.0 THE 1998 SREP PROGRAM

SELECTION DATA: A total of 490 faculty members (SFRP Associates) and 202 graduate students (GSRP associates) applied to participate in the 1998 Summer Research Program. From these applicants 188 SFRPs and 98 GSRPs were selected. The education level of those selected was as follows:

1997 SRP Associates, by Degree			
SFRP		GSRP	
PHD	MS	MS	BS
184	6	2	53

Of the participants in the 1997 Summer Research Program 90 percent of SFRPs and 13 percent of GSRPs submitted proposals for the SREP. One undred and thirty-two proposals from SFRPs and seventeen from GSRPs were selected for funding, which equates to a selection rate of 54% of the SFRP proposals and of 34% for GSRP proposals.

1998 SREP: Proposals Submitted vs. Proposals Selected			
	Summer 1997 Participants	Submitted SREP Proposals	SREPs Funded
SFRP	188	132	20
GSRP	98	17	4
TOTAL	286	149	24

The funding was provided as follows:

Contractual slots funded by AFOSR	18
Laboratory funded	22
Total	40

Twelve HBCU/MI associates from the 1997 summer program submitted SREP proposals; six were selected (none were lab-funded; all were funded by additional AFOSR funds).

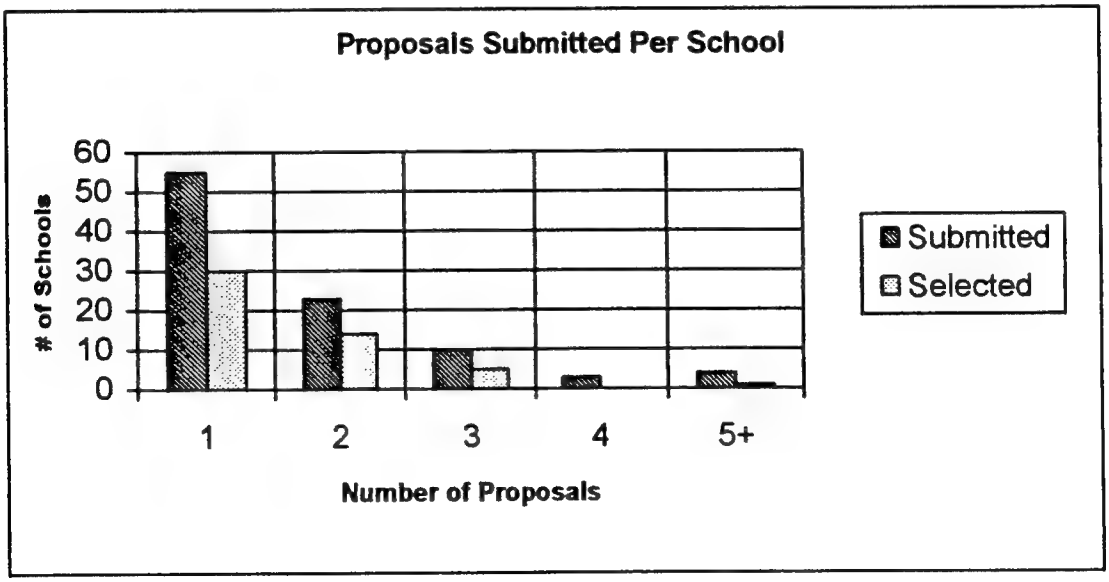
Proposals Submitted and Selected, by Laboratory		
	Applied	Selected
Armstrong Research Site	9	3
Air Logistic Centers	31	5
Arnold Engineering Development Center	2	1
Phillips Research Site	30	10
Rome Research Site	29	12
Wilford Hall Medical Center	1	0
Wright Research Site	47	9
TOTAL	149	40

Note: Armstrong Research Site funded 1 SREP; Phillips Research Site funded 6; Rome Research Site funded 9; Wright Research Site funded 6.

The 125 1997 Summer Research Program participants represented 60 institutions.

Institutions Represented on the 1997 SRP and 1998 SREP		
Number of schools represented in the Summer 97 Program	Number of schools represented in submitted proposals	Number of schools represented in Funded Proposals
125	110	55

Thirty schools had more than one participant submitting proposals.



The selection rate for the 65 schools submitting 1 proposal (68%) was better than those submitting 2 proposals (61%), 3 proposals (50%), 4 proposals (0%) or 5+ proposals (25%). The 4 schools that submitted 5+ proposals accounted for 30 (15%) of the 149 proposals submitted.

Of the 149 proposals submitted, 130 offered institution cost sharing. Of the funded proposals which offered cost sharing, the minimum cost share was $3046.00, the maximum was $39,261.00 with an average cost share of $11,069.21.

Proposals and Institution Cost Sharing		
	Proposals Submitted	Proposals Funded
With cost sharing	117	32
Without cost sharing	32	8
Total	149	40

The SREP participants were residents of 31 different states. Number of states represented at each laboratory were:

States Represented, by Proposals Submitted/Selected per Laboratory		
	Proposals Submitted	Proposals Funded
Armstrong Laboratory	31	5
Air Logistic Centers	9	3
Arnold Engineering Development Center	2	1
Phillips Laboratory	30	10
Rome Laboratory	29	12
Wilford Hall Medical Center	1	0
Wright Laboratory	47	9

Nine of the 1997 SREP Principal Investigators also participated in the 1998 SREP.

ADMINISTRATIVE EVALUATION: The administrative quality of the SREP associates' final reports was satisfactory. Most complied with the formatting and other instructions provided to them by RDL. Thirty-seveb final reports have been received and are included in this report. The subcontracts were funded by $992,855.00 of Air Force money. Institution cost sharing totaled $354.215.00.

TECHNICAL EVALUATION: The form used for the technical evaluation is provided as Appendix 2. Thirty-five evaluation reports were received. Participants by laboratory versus evaluations submitted is shown below:

	Participants	Evaluations	Percent
Armstrong Laboratory	5	4	95.2
Air Logistic Centers	3	3	100
Arnold Engineering Development Center	1	1	100
Phillips Laboratory	10	10	100
Rome Laboratory	12	12	100
Wright Laboratory	9	5	91.9
Total	40	35	95.0

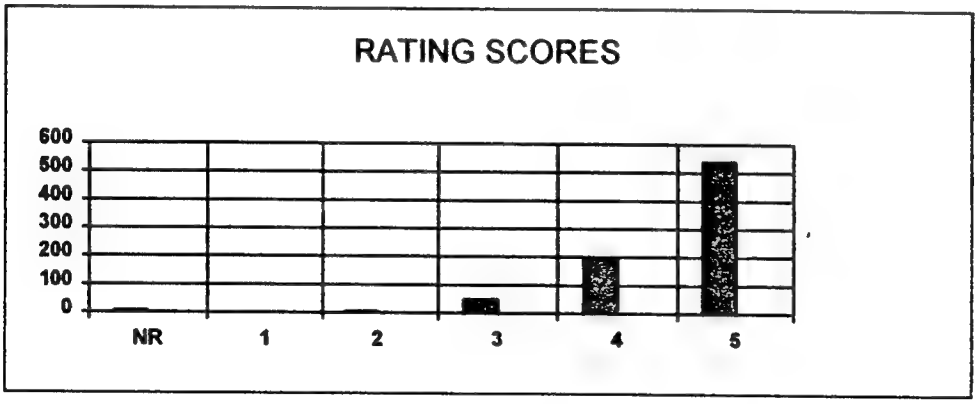
Notes:

1: Research on four of the final reports was incomplete as of press time so there aren't any technical evaluations on them to process, yet. Percent complete is based upon 20/21=95.2%

2: One technical evaluation was not completed because one of the final reports was incomplete as of press time. Percent complete is based upon 18/18=100%

The number of evaluations submitted for the 1998 SREP (95.0%) shows a marked improvement over the 1997 SREP submittals (65%).

PROGRAM EVALUATION: Each laboratory focal point evaluated ten areas (see Appendix 2) with a rating from one (lowest) to five (highest). The distribution of ratings was as follows:

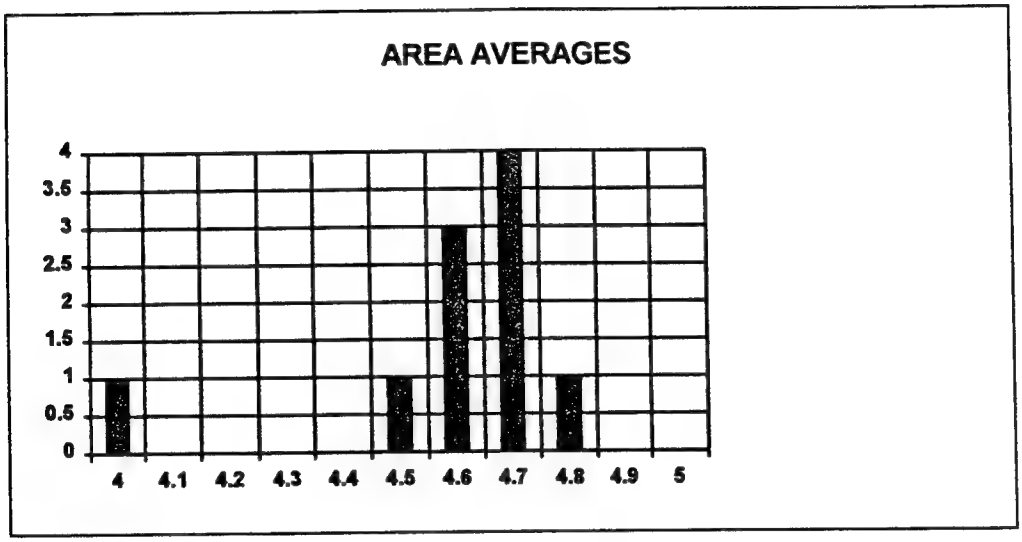


Rating	Not Rated	1	2	3	4	5
# Responses	7	1	7	62 (6%)	226 (25%)	617 (67%)

The 8 low ratings (one 1 and seven 2's) were for question 5 (one 2) "The USAF should continue to pursue the research in this SREP report" and question 10 (one 1 and six 2's) "The one-year period for complete SREP research is about right", in addition over 30% of the threes (20 of 62) were for question ten. The average rating by question was:

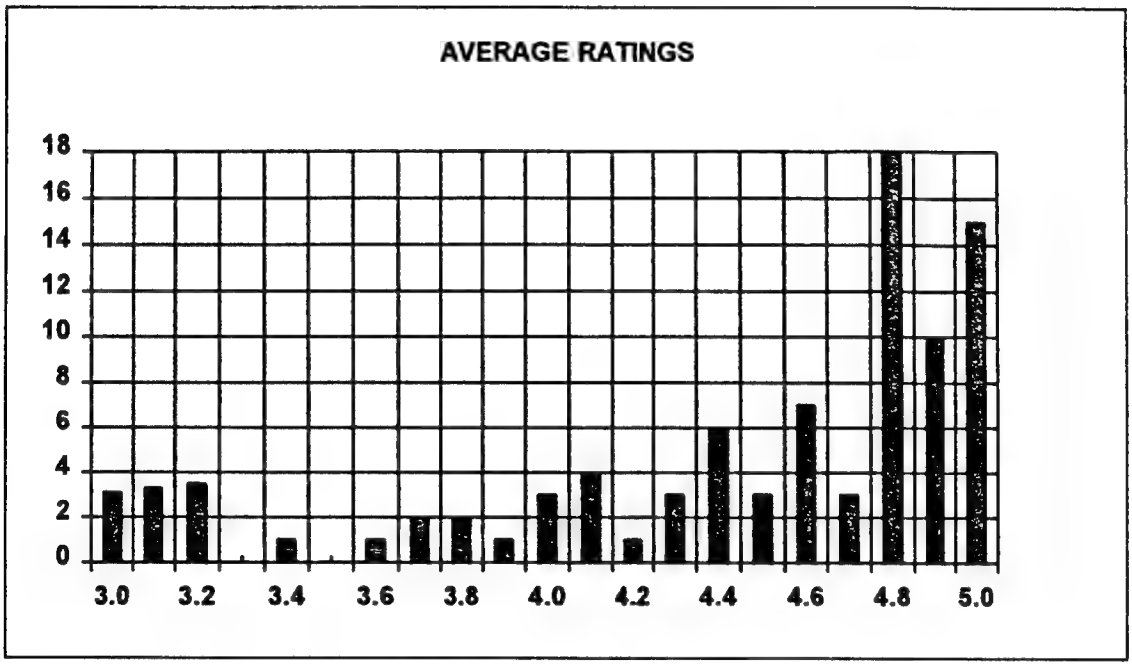
Question	1	2	3	4	5	6	7	8	9	10
Average	4.6	4.6	4.7	4.7	4.6	4.7	4.8	4.5	4.6	4.0

The distribution of the averages was:



Area 10 "the one-year period for complete SREP research is about right" had the lowest average rating (4.1). The overall average across all factors was 4.6 with a small sample standard deviation of 0.2. The average rating for area 10 (4.1) is approximately three sigma lower than the overall average (4.6) indicating that a significant number of the evaluators feel that a period of other than one year should be available for complete SREP research.

The average ratings ranged from 3.4 to 5.0. The overall average for those reports that were evaluated was 4.6. Since the distribution of the ratings is not a normal distribution the average of 4.6 is misleading. In fact over half of the reports received an average rating of 4.8 or higher. The distribution of the average report ratings is as shown:



It is clear from the high ratings that the laboratories place a high value on AFOSR's Summer Research Extension Programs.

3.0 SUBCONTRACTS SUMMARY

Table 1 provides a summary of the SREP subcontracts. The individual reports are published in volumes as shown:

Laboratory	Volume
Armstrong Research Site	1
Arnold Engineering Development Center	5
Air Logistic Centers	5
Phillips Research Site	2
Rome Research Site	3
Wright Research Site	4

SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Chubb , Gerald **Industrial Engineering** **Ohio State University, Columbus, OH**	**PhD** 98-0829	**AL/HR** Scoring Pilot Performance of Basic Flight Manuevers	**01/01/98 12/31/98**	**$25000.00**	**$0.00**
Foy , Brent **Medical Physics** **Wright State University, Dayton, OH**	**PhD** 98-0828	**AL/OE** Development & Validation of a Physiologically-Based Kinetic Model of Perfused	**01/01/98 12/31/98**	**$25000.00**	**$11278.00**
Lance , Charles **Psychology** **Univ of Georgia Res Foundation, Athens, GA**	**PhD** 98-0842	**AL/HR** Extension of Job Performance Measurement Tech to the Development of a Prototype	**01/01/98 12/31/98**	**$24989.00**	**$0.00**
Woehr , David **Department of Psychology** **Texas A & M Univ-College Station, College**	**PhD** 98-0802	**AL/HR** Validation of The Multidimensional work ethic profile (MWEP) as a screening too	**01/01/98 12/31/98**	**$25000.00**	**$11508.00**
Collins , Frank **Mechanical Engineering** **Tennessee Univ Space Institute, Tullahoma, TN**	**PhD** 98-0807	**AEDC/E** Monte Carlo Computation of SpeciesSepaaration by a Conical Skimmer in Hypersonic	**01/01/98 12/31/98**	**$25000.00**	**$16104.00**
Whaley , Paul **Mechanical Engineering** **Oklahoma Christian Univ of Science & Art,**	**PhD** 98-0820	**ALC/OC** Probabilistic Analysis of Residual Strength in Corroded and Uncorroded Aging Air	**01/01/98 12/31/98**	**$23351.00**	**$3046.00**
Balas , Mark **Applied Math** **Univ of Colorado at Boulder, Boulder, CO**	**PhD** 98-0816	**PL/SX** Non-Linear Adaptive Control for a Precision Deployable Structure with White ligh	**01/01/98 12/31/98**	**$25000.00**	**$0.00**
Duric , Neb **Astrophysics** **University of New Mexico, Albuquerque, NM**	**PhD** 98-0808	**PL/LI** Image Recovery Using Phase Diversity	**01/01/98 12/31/98**	**$25000.00**	**$5777.00**
Hanson , George **Electrical Engineering** **Univ of Wisconsin - Milwaukee, Milwaukee, WI**	**PhD** 98-0811	**PL/WS** Perturbation Analysis of the Natural Frequencies Targets in Inhomogeneous Media	**01/01/98 12/31/98**	**$25000.00**	**$23250.00**
Jeffs , Brian **Electrical Engineering** **Brigham Young University, Provo, UT**	**PhD** 98-0813	**PL/LI** Bayesian Restoration of Space object Images From Adaptive Optics Data with unkno	**01/01/98 12/31/98**	**$25000.00**	**$19177.00**
Kar , Aravinda **Engineering** **University of Central Florida, Orlando, FL**	**PhD** 98-0812	**PL/LI** Effects of Vapor-Plasma Layer on Thick-Section Cutting and Calculation of Modes	**01/01/98 12/31/98**	**$25000.00**	**$5414.00**
Leo , Donald **Mechanical & Aerospace** **University of Toledo, Toledo, OH**	**PhD** 98-0810	**PL/VT** Adaptive vibration suppression for autonomous Control Systems	**01/01/98 09/30/98**	**$24964.00**	**$9628.00**
Liu , Hanli **Physics** **Univ of Texas at Arlington, Arlington, TX**	**PhD** 98-0814	**PL/LI** Continuous-Wave Approach to 3-D Imaging through Turbid media w/a Single Planar M	**01/01/98 12/31/98**	**$25000.00**	**$11000.00**
Bienfang , Joshua **Physics** **University of New Mexico, Albuquerque, NM**	**BS** 98-0815	**PL/LI** Optical Clocks Based on Diode Lasers	**01/01/98 12/31/98**	**$24994.00**	**$0.00**
Paulson , Eric **Engineering/Physics** **Univ of Colorado at Boulder, Boulder, CO**	**BS** 98-0837	**PL/RK** Optimization & Analysis of a Waverider Vehicle for Global Spaceplane Trajectorie	**01/01/98 12/31/98**	**$25000.00**	**$7794.00**

SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Stephens II , Kenneth	**MA**	**PL/WS**	**01/01/98 12/31/98**	**$25000.00**	**$16764.00**
University of North Texas, Denton, TX	98-0809	Simulation of an Explosively Formed Fuse Using MACH 2			
Barjaktarovic , Milica	**PhD**	**RL/IW**	**01/01/98 12/31/98**	**$24976.00**	**$3158.00**
Electrical Engineering	98-0824	Specification and Verification of SDN.701 MSP			
Wilkes University, Wilkes Barre, PA		Functions and Missi Crypto Functio			
Batalama , Stella	**PhD**	**RL/C3**	**01/01/98 12/31/98**	**$25000.00**	**$5600.00**
EE	98-0823	Robust Spread Spectrum Communications:Adaptive			
SUNY Buffalo, Buffalo, NY		Interference Mitigation Technique			
Bourbakis , Nikolaos	**PhD**	**RL/IR**	**01/01/98 12/31/98**	**$25000.00**	**$22723.00**
Computer Science & Engr	98-0832	hierarchical-Adaptive Image Segmentation			
SUNY Binghamton, Binghamton, NY					
Dasigi , Venugopala	**PhD**	**RL/C3**	**01/01/98 12/31/98**	**$25000.00**	**$4000.00**
Computer Science	98-0830	Information Fusion w/Multiple Feature Extractors			
Southern Polytechnic State Univ, Marietta, GA		for automatic Text Classificati			
Eckert , Richard	**PhD**	**RL/C3**	**01/01/98 12/31/98**	**$25000.00**	**$39261.00**
Physics	98-0825	The Interactive Learning Wall; A PC-Based,			
SUNY Binghamton, Binghamton, NY		Deployable Data Wall for Use in a Co			
Lin , Kuo-Chi	**PhD**	**RL/IR**	**01/01/98 12/31/98**	**$25000.00**	**$0.00**
Aerospace Engineering	98-0822	Web-Based Distributed Simulation			
University of Central Florida, Orlando, FL					
Pados , Dimitrios	**PhD**	**RL/OC**	**01/01/98 12/31/98**	**$25000.00**	**$5600.00**
Dept. of Electrical /Computer Eng.	98-0818	Adaptive Array Radars and Joint Space-Time			
State Univ. of New York Buffalo, Buffalo, NY		Auxiliary Verctor Filtering			
Panda , Brajendra	**PhD**	**RL/CA**	**01/01/98 12/31/98**	**$25000.00**	**$7113.00**
Computer Science	98-0821	Information Warfare: Design of an Efficient Log			
University of North Dakota, Grand Forks, ND		Management Method to Aid In Dat			
Pittarelli , Michael	**PhD**	**RL/C3**	**01/01/98 12/31/98**	**$24998.00**	**$0.00**
Systems Science	98-0827	Complexity of Detecting and content-driven			
SUNY OF Tech Utica, Utica, NY		methods for resolving database incons			
Schmalz , Mark	**PhD**	**RL/IR**	**01/01/98 12/31/98**	**$24619.00**	**$0.00**
Dept of Computer & Info Science	98-0831	Errors Inherent in 3D Target Reconstruction from			
University of Florida, Gainesville, FL		Multiple Airborne Images			
Ye , Nong	**PhD**	**RL/CA**	**01/01/98 12/31/98**	**$25000.00**	**$5000.00**
Industrial Engineering	98-0826	Model-Based Assessment of Campaign Plan			
Arizona State University, Tempe, AZ		Performance under Uncertainty			
Bradley , Parker	**BS**	**RL/IR**	**01/01/98 12/31/98**	**$25000.00**	**$0.00**
Physics	98-0834	Development of User-Friendly Comp Environment			
Syracuse University, Syracuse, NY		for Blind Source Separation Studie			
Kumar , Devendra	**PhD**	**ALC/SA**	**01/01/98 12/31/98**	**$25000.00**	**$11362.00**
Computer Science	98-0805	Further Development of a Simpler, Multiversion			
CUNY-City College, New York, NY		concurrency Control Protocol for			
Chow , Joe	**PhD**	**ALC/W**	**01/01/98 12/31/98**	**$25000.00**	**$5360.00**
Mechanical Engineering	98-0806	An Automated 3-D Surface Model Creation Module			
Florida International Univ, Miami, FL		for Laser Scanned Point Data			

SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Beecken , Brian **Physics** **Bethel College, St. Paul, MN**	**PhD** 98-0804	**WL/MN** Development of a statistical Model predicting the impact of a scene projector's	**01/01/98 12/31/98**	**$19986.00**	**$3997.00**
Beggs , John **Electrical Engineering** **Mississippi State University, Mississippi State,**	**PhD** 98-0817	**WL/FI** Implementation of an Optimization Algorithm in Electromagnentics for Radar Absor	**01/01/98 12/31/98**	**$25000.00**	**$25174.00**
Bhatnagar , Raj **Computer Science** **University of Cincinnati, Cincinnati, OH**	**PhD** 98-0819	**WL/AA** Analysis of Intra-Class Variability & synthetic Target Models for Use in ATR	**01/01/98 09/30/98**	**$25000.00**	**$17488.00**
Blaisdell , Gregory **Mechanical Engineering** **Purdue University, West Lafayette, IN**	**PhD** 98-0839	**WL/FI** Validation of a Large Eddy Simulation Code & Development of Commuting Filters	**01/01/98 12/31/98**	**$25000.00**	**$11844.00**
Douglass , John **Zoology** **University of Arizona, Tucson, AZ**	**PhD** 98-0803	**WL/MN** Roles of Matched Filtering and Coarse in Insect Visual Processing	**01/01/98 12/31/98**	**$25000.00**	**$3719.00**
Hosford , William **Mettalurgy** **Univ of Michigan, Ann Arbor, MI**	**PHD** 98-0840	**WL/MN** Prediction of Compression Textures in Tantalum Using a Pencil-Glide Computer Mod	**01/01/98 12/31/98**	**$25000.00**	**$5000.00**
Pan , Yi **Computer Science** **University of Dayton, Dayton, OH**	**PhD** 98-0838	**WL/FI** Parallelization of Time-Dependent Maxwell Equations Using High Perform. Fortran	**01/01/98 12/31/98**	**$25000.00**	**$9486.00**
Pochiraju , Kishore **Mechanical Engineering** **Stevens Inst of Technology, Hoboken, NJ**	**PhD** 98-0833	**WL/ML** A Hybrid Variational-Asymptotic Method for the Analysis of MicroMechanical Damag	**01/01/98 12/31/98**	**$25000.00**	**$9625.00**
Shtessel , Yuri **Electrical Engineering** **Univ of Alabama at Huntsville, Huntsville, AL**	**PhD** 98-0841	**WL/FI** Continuous Sliding Mode Control Approach for Addressing actutor Deflection and	**01/01/98 12/31/98**	**$25000.00**	**$4969.00**
Starzyk , Janusz **Electrical Engineering** **Ohio University, Athens, OH**	**PhD** 98-0801	**WL/AA** Feature Selection for Automatic Target Recognition:Mutual Info & Stat Tech	**01/01/98 12/31/98**	**$24978.00**	**$12996.00**

APPENDIX 1:

SAMPLE SREP SUBCONTRACT

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
1998 SUMMER RESEARCH EXTENSION PROGRAM
SUBCONTRACT 98-0812

BETWEEN

Research & Development Laboratories
5800 Uplander Way
Culver City, CA 90230-6608

AND

University of Central Florida
Office of Sponsored Research/ Admin#423
4000 Central Florida Blvd.
Orlando, FL 32816-0150

REFERENCE: Summer Research Extension Program Proposal 97-0018

Start Date: 01/01/98 End Date: 12/31/98

Proposal Amount: $25000.0
Proposal Title:
Effects of Vapor-Plasma Layer on Thick-Section Cutting and Calculation of Modes

(1) PRINCIPAL INVESTIGATOR:

DR Aravinda Kar
CREOL
University of Central Florida
Orlando, FL 32816-2700

(2) UNITED STATES AFOSR CONTRACT NUMBER: F49620-93-C-0063

(3) CATALOG OF FEDERAL DOMESTIC ASSISTANCE NUMBER (CFDA): 12.800
PROJECT TITLE: AIR FORCE DEFENCE RESEARCH SOURCES PROGRAM

(4) ATTACHMENTS
1. **REPORT OF INVENTIONS AND SUBCONTRACT**
2. **CONTRACT CLAUSES**
3. **FINAL REPORT INSTRUCTIONS**

***** SIGN SREP SUBCONTRACT AND RETURN TO RDL *****

1. BACKGROUND: Research & Development Laboratories (RDL) is under contract (F49620-93-C-0063) to the United States Air Force to administer the Summer Research Program (SRP), sponsored by the Air Force Office of Scientific Research (AFOSR), Bolling Air Force Base, D.C. Under the SRP, a selected number of college faculty members and graduate students spend part of the summer conducting research in Air Force laboratories. After completion of the summer tour participants may submit, through their home institutions. proposals for follow-on research. The follow-on research is known as the Summer Research Extension Program (SREP). Approximately 61 SREP proposals annually will be selected by the Air Force for funding of up to $25,000; shared funding by the academic institution is encouraged. SREP efforts selected for funding are administered by RDL through subcontracts with the institutions. This subcontract represents an agreement between RDL and the institution herein designated in Section 5 below.

2. RDL PAYMENTS: RDL will provide the following payments to SREP institutions:
   - 80 percent of the negotiated SREP dollar amount at the start of the SREP research period.
   - The remainder of the funds within 30 days after receipt at RDL of the acceptable written final report for the SREP research.

3. INSTITUTION'S RESPONSIBILITIES: As a subcontractor to RDL, the institution designated on the title page will:

a. Assure that the research performed and the resources utilized adhere to those defined in the SREP proposal.

b. Provide the level and amounts of institutional support specified in the SREP proposal..

c. Notify RDL as soon as possible, but not later than 30 days, of any changes in 3a or 3b above, or any change to the assignment or amount of participation of the Principal Investigator designated on the title page.

d. Assure that the research is completed and the final report is delivered to RDL not later than twelve months from the effective date of this subcontract, but no later than December 31, 1998. The effective date of the subcontract is one week after the date that the institution's contracting representative signs this subcontract, but no later than January 15, 1998.

e. Assure that the final report is submitted in accordance with Attachment 3.

f. Agree that any release of information relating to this subcontract (news releases, articles, manuscripts, brochures, advertisements, still and motion pictures, speeches, trade associations meetings, symposia, etc.) will include a statement that the project or effort depicted was or is sponsored by: Air Force Office of Scientific Research, Bolling AFB, D.C.

g. Notify RDL of inventions or patents claimed as the result of this research as specified in Attachment 1.

h. RDL is required by the prime contract to flow down patent rights and technical data requirements to this subcontract. Attachment 2 to this subcontract

contains a list of contract clauses incorporated by reference in the prime contract.

4. All notices to RDL shall be addressed to:

RDL AFOSR Program Office
5800 Uplander Way
Culver City, CA 90230-6609

5. By their signatures below, the parties agree to provisions of this subcontract.

Abe Sopher RDL Contracts Manager	Signature of Institution Contracting Official
	Typed/Printed Name
Date	Title
	Institution
	Date/Phone

ATTACHMENT 2
CONTRACT CLAUSES

This contract incorporates by reference the following clauses of the Federal Acquisition Regulations (FAR), with the same force and effect as if they were given in full text. Upon request, the Contracting Officer or RDL will make their full text available (FAR 52.252-2).

FAR CLAUSES	TITLE AND DATE
52.202-1	DEFINITIONS
52.203-3	GRATUITIES
52.203-5	COVENANT AGAINST CONTINGENT FEES
52.203-6	RESTRICTIONS ON SUBCONTRACTOR SALES TO THE GOVERNMENT
52.203-7	ANTI-KICKBACK PROCEDURES
52.203-8	CANCELLATION, RECISSION, AND RECOVERY OF FUNDS FOR ILLEGAL OR IMPROPER ACTIVITY
52.203-10	PRICE OR FEE ADJUSTMENT FOR ILLEGAL OR IMPROPER ACTIVITY
52.203-12	LIMITATION ON PAYMENTS TO INFLUENCE CERTAIN FEDERAL TRANSACTIONS
52.204-2	SECURITY REQUIREMENTS
52.209-6	PROTECTING THE GOVERNMENT'S INTEREST WHEN SUBCONTRACTING WITH CONTRACTORS DEBARRED, SUSPENDED, OR PROPOSED FOR DEBARMENT
52.212-8	DEFENSE PRIORITY AND ALLOCATION REQUIREMENTS
52.215-2	AUDIT AND RECORDS - NEGOTIATION
52.215-10	PRICE REDUCTION FOR DEFECTIVE COST OR PRICING DATA

52.215-12	SUBCONTRACTOR COST OR PRICING DATA
52.215-14	INTEGRITY OF UNIT PRICES
52.215-8	ORDER OF PRECEDENCE
52.215.18	REVERSION OR ADJUSTMENT OF PLANS FOR POSTRETIREMENT BENEFITS OTHER THAN PENSIONS
52.222-3	CONVICT LABOR
52.222-26	EQUAL OPPORTUNITY
52.222-35	AFFIRMATIVE ACTION FOR SPECIAL DISABLED AND VIETNAM ERA VETERANS
52.222-36	AFFIRMATIVE ACTION FOR HANDICAPPED WORKERS
52.222-37	EMPLOYMENT REPORTS ON SPECIAL DISABLED VETERAN AND VETERANS OF THE VIETNAM ERA
52.223-2	CLEAN AIR AND WATER
52.223-6	DRUG-FREE WORKPLACE
52.224-1	PRIVACY ACT NOTIFICATION
52.224-2	PRIVACY ACT
52.225-13	RESTRICTIONS ON CONTRACTING WITH SANCTIONED PERSONS
52.227-1	ALT. I - AUTHORIZATION AND CONSENT
52.227-2	NOTICE AND ASSISTANCE REGARDING PATIENT AND COPYRIGHT INFRINGEMENT

52.227-10	FILING OF PATENT APPLICATIONS - CLASSIFIED SUBJECT MATTER
52.227-11	PATENT RIGHTS - RETENTION BY THE CONTRACTOR (SHORT FORM)
52.228-7	INSURANCE - LIABILITY TO THIRD PERSONS
52.230-5	COST ACCOUNTING STANDARDS - EDUCATIONAL INSTRUCTIONS
52.232-23	ALT. I - ASSIGNMENT OF CLAIMS
52.233-1	DISPUTES
52.233-3	ALT. I - PROTEST AFTER AWARD
52.237-3	CONTINUITY OF SERVICES
52.246-25	LIMITATION OF LIABILITY - SERVICES
52.247-63	PREFERENCE FOR U.S. - FLAG AIR CARRIERS
52.249-5	TERMINATION FOR CONVENIENCE OF THE GOVERNMENT (EDUCATIONAL AND OTHER NONPROFIT INSTITUTIONS)
52.249-14	EXCUSABLE DELAYS
52.251-1	GOVERNMENT SUPPLY SOURCES

DOD FAR CLAUSES	DESCRIPTION
252.203-7001	SPECIAL PROHIBITION ON EMPLOYMENT
252.215-7000	PRICING ADJUSTMENTS
252.233-7004	DRUG FREE WORKPLACE (APPLIES TO SUBCONTRACTS WHERE THERE IS ACCESS TO CLASSIFIED INFORMATION)
252.225-7001	BUY AMERICAN ACT AND BALANCE OF PAYMENTS PROGRAM
252.225-7002	QUALIFYING COUNTRY SOURCES AS SUBCONTRACTS
252.227-7013	RIGHTS IN TECHNICAL DATA - NONCOMMERCIAL ITEMS
252.227-7030	TECHNICAL DATA - WITHOLDING PAYMENT
252.227-7037	VALIDATION OF RESTRICTIVE MARKINGS ON TECHNICAL DATA
252.231-7000	SUPPLEMENTAL COST PRINCIPLES
252.232-7006	REDUCTIONS OR SUSPENSION OF CONTRACT PAYMENTS UPON FINDING OF FRAUD

APPENDIX 2:

SAMPLE TECHNICAL EVALUATION FORM

SUMMER RESEARCH EXTENSION PROGRAM
TECHNICAL EVALUATION

SREP No: 98-0810
Principal Investigator: DR Donald Leo
University of Toledo

Circle the rating level number, 1 (low) through 5 (high), you feel best evaluate each statement and return the completed form to RDL by fax of mail to:

RDL
Attn: SREP Tech Evals
5800 Uplander Way
Culver City, CA 90230-6608
(310)216-5940 or (800)677-1363

1.	This SREP report has a high level of technical merit.	1 2 3 4 5
2.	The SREP program is important to accomplishing the lab's mission.	1 2 3 4 5
3.	This SREP report accomplished what the associate's proposal promised.	1 2 3 4 5
4.	This SREP report addresses area(s) important to the USAF.	1 2 3 4 5
5.	The USAF should continue to pursue the research in this SREP report.	1 2 3 4 5
6.	The USAF should maintain research relationships with this SREP associate.	1 2 3 4 5
7.	The money spent on this SREP effort was well worth it.	1 2 3 4 5
8.	This SREP report is well organized and well written.	1 2 3 4 5
9.	I'll be eager to be a focal point for summer and SREP associates in the future.	1 2 3 4 5
10.	The one-year period for complete SREP research is about right.	1 2 3 4 5

---

11. If you could change any one thing about the SREP program, what would you change:

_____

_____

_____

_____

12. What do you definitely NOT change about the SREP program?

_____

_____

_____

_____

PLEASE USE THE BACK FOR ANY OTHER COMMENTS

Laboratory: Phillips Laboratory
Lab Focal Point: Capt Jeanne Sullivan
Office Symbol: AFRL/VSDV Phone: (505)846-2069

APPLICATION OF QUATERNIONS TO COMPUTE MODEL MOTIONS FOR VIRTUAL FLIGHT TESTING

Frank G. Collins
Professor
Department of Mechanical and Aerospace Engineering and Engineering Mechanics

The University of Tennessee Space Institute
Tullahoma, TN 37388-9700

Final Report for:
Summer Research Extension Program
Arnold Engineering Development Center

Sponsored by:
Air Force Office of Scientific Research
Boling Air Force Base, DC

and

Arnold Engineering Development Center

March 1999

APPLICATION OF QUATERNIONS TO COMPUTE MODEL MOTIONS FOR VIRTUAL FLIGHT TESTING

Frank G. Collins
Professor
Department of Mechanical and Aerospace Engineering and Engineering Mechanics
The University of Tennessee Space Institute

Abstract

Present wind tunnel testing allows for the measurement of static forces and moments on a model of an air vehicle in a fixed orientation relative to the air stream. Dynamic derivatives for the full-scale vehicle are then predicted from the static wind tunnel measurements and eventually from costly full-scale flight testing. Nonlinear, unsteady, high angle-of-attack aerodynamic effects cannot be determined by this wind tunnel method. The Virtual Flight Testing (VFT) technique has been proposed as a technique that would allow the wind tunnel to make measurements of dynamic aerodynamic effects. A requirement of VFT is the rapid and accurate computation of the kinematics of the model of the flight vehicle while it is undergoing a rapid change in orientation in the wind tunnel. The computational method must also be free of singularities. The quaternion method has been shown in the past to be the most useful method for meeting these requirements. This method is described in this report and compared to the rotational matrix and Euler angle methods. The equations of motion for an unconstrained flight vehicle are derived as a example of the use of the method. Detailed equations for the various transformations are given.

APPLICATION OF QUATERNIONS TO COMPUTE MODEL MOTIONS FOR VIRTUAL FLIGHT TESTING

Frank G. Collins

Introduction

Present wind tunnel testing methodology yields static force and moment data for a flight vehicle, requiring flight testing to measure the dynamic response of the vehicle. Flight testing is extremely expensive to perform. Virtual Flight Testing (VFT) proposes to allow for the limited dynamic testing of flight vehicles in wind tunnels, thus reducing the expense of later flight testing (Gebert, Kelly and Lopez (1998)).

The Virtual Flight Testing method would attach the flight test model to a support system that will allow the model to have free rotational motion and limited pitch and yaw motion. The model will be allowed to respond to the aerodynamic forces and moments in this restricted fashion and will be flying under the control of its own autopilot.

The model and autopilot will be part of a hardware-in-the-loop system which must include a computer computation of the kinematics of motion sensed by the on-board accelerometers and rate gyros. The kinematical computations must be performed with as much speed and accuracy as possible, and without the presence of any possible singularities. These considerations are particularly important for testing of the newer classes of missiles which must maneuver to hit a target which is behind the aircraft that launched the missile. Such missiles require pitch/yaw rates of 300°-600°/sec and angles-of-attack to 70°. The quaternion method for computing the rotational orientation changes of an aerodynamic body has previously been found to be 30% faster that the traditional rotational matrices method (Robinson(1958)). It is for this reason that the present report emphasizes the use of the quaternion method.

Finite rotations in space are described by 3×3 rotational transformation matrices. Traditionally the three Euler angles are used to describe model orientations. However, they are computationally very time consuming and possess singularities at certain angles. The rotational matrix does not possess any singularities but it has nine parameters with six constraints. The minimum number of parameters that can be used to describe finite rotations in space without singularity is four (Chou (1992)).

The most useful of these four parameter representations is the Euler parameters, which are unit quaternions. The quaternion representation of finite body rotation in three-dimensional space will be fully described in this report. The application of quaternions to the description of the wind tunnel model and model support orientation will be given. The equations of motion for the model plus the equation for the update of the quaternion that describies the continuous change of orientation of the rigid body will be derived. Additional aspects of the application of quaternions that will not be covered in this report are given by Chou (1992) and Kuipers (1999).

Finite Rotations in Three-Dimensions

A discussion of rigid body motion requires the introduction of coordinate systems against which the body rotation can be measured as a change in orientation of two sets of respective coordinate axes. See Figure 1. In this report two primary coordinate systems will be used, an earth-based system (approximately inertial) and a body-fixed system. The body-fixed system (or systems) will be further defined when needed. The earth and body coordinate systems are assumed to be initially aligned. An *active* rotation is one that rotates the coordinate system from the earth-based system to the body-fixed system. Active and its opposite, passive, rotations will be discussed in detail later.

The following facts about rigid body motion can be easily developed (Katz (1997)):

1. Rigid body motion can be considered to be a mapping of the location of the body from one position to another. The mapping is such that two distinct points in the body cannot be mapped into the same point.

2. A rigid body motion has an inverse. If the first motion can be described by $\mathbf{x} \rightarrow \mathbf{x}'$ then an inverse motion $\mathbf{x}' \rightarrow \mathbf{x}$ is possible.

3. There is a null (identity) motion such that $\mathbf{x} \rightarrow \mathbf{x}$.

4. Rigid body motions can be combined to give a single operation taking the body from the initial to the final orientation. If $\mathbf{x} \rightarrow \mathbf{x}'$ and $\mathbf{x}' \rightarrow \mathbf{x}''$, then there is a resultant motion $\mathbf{x} \rightarrow \mathbf{x}''$.

These properties demonstrate that finite rotation of a rigid body forms a *group*. A group is defined as a collection of objects G which has an operation '*' which assigns to any elements x and y in the set an element x*y which also belongs to the set G. The operation must satisfy three laws: 1) for any x, y, and z in G, x*(y*z) = (x*y)*z; 2) there is an identity element I in G such that I*x = x = x*I for any x in G; and 3) for any x in G there is an inverse x′ in G such that x*x′ = I = x′*x. For rotation of a finite body, the null motion is the identity group element and the operation of combining motions is the group operation '*'.

Rotation Matrices

Finite rotations of a solid body in three-dimensional space can be described by the rotation matrix $\underline{\underline{R}}$. Rigid body motion can be split into translation plus rotation about a point. The axis of rotation passes through this point. Take this point to be the origin of the coordinate system so we are interested in rotations about the origin. Then rotations about this origin can be described as follows. Take the position vector from this origin to a point in the body to be $\mathbf{x}$. Upon rotation of the rigid body this position vector is transformed in the vector $\mathbf{x}'$, as measured in an inertial frame. The transformed position

vector is obtained by operating upon (matrix multiplication) $\mathbf{x}$ by the rotation matrix $\underline{\underline{R}}$, i. e.,

$$\mathbf{x}' = \underline{\underline{R}}\mathbf{x} \tag{1}$$

where both position vectors are taken as column vectors. $\mathbf{x}^T$, the transform of $\mathbf{x}$, designates a row vector. In terms of row vectors the rotation can be written as

$$\mathbf{x}'^T = \mathbf{x}\underline{\underline{R}}^T \tag{2}$$

where $\underline{\underline{R}}^T$ is the transpose of the rotation matrix. If the rigid body first undergoes a rotation described by the rotation matrix $\underline{\underline{R}}^1$ followed by a second rotation described by the rotation matrix $\underline{\underline{R}}^2$ then the resultant orientation can be obtained by a single rotation matrix given by

$$\underline{\underline{R}}' = \underline{\underline{R}}^2\underline{\underline{R}}^1 \tag{3}$$

where matrix multiplication is indicated. Starting with a *reference orientation,* each orientation may be represented by the rotation that takes the body from the reference orientation to its current orientation. This may be accomplished by a single resultant rotation matrix $\underline{\underline{R}}'$, as shown above.

The rotation matrix $\underline{\underline{R}}$ possesses the following properties (Katz (1997) and Goldstein (1950)):

1. $\underline{\underline{R}}$ is a linear operator;
2. $\underline{\underline{R}}$ is an orthogonal matrix, i. e., its inverse is equal to its transpose,

$$\underline{\underline{R}}^T = \underline{\underline{R}}^{-1}; \tag{4}$$

3. Every orthogonal matrix induces a rigid body rotation;
4. The transpose of the rotation matrix, $\underline{\underline{R}}^T$, is an orthogonal matrix;
5. The resultant rotation matrix generated by successive rotations, $\underline{\underline{R}}'$, is orthogonal;
6. Writing the position vector in terms of its components as

$$\mathbf{x} = x_i\mathbf{e}_i \tag{5}$$

where $\mathbf{e}_i$ are unit vectors in the three mutually perpendicular directions, then the components, $\underline{\underline{R}}_{ij}$, of the rotation matrix for the transformation from $\mathbf{x}$ to $\mathbf{x}'$ is given by the scalar product

$$\underline{\underline{R}}_{ij} = \mathbf{e}_i \cdot \mathbf{e}'_j; \tag{6}$$

these components are equal to the cosine of the angle between the x_i and x_j' axes. The subscript i designates a row of the matrix and j a column, i, j = 1, 2, 3;

7. The determinant of the square of the rotation matrix is equal to one, i. e.

$$\left|\underline{\underline{R}}\right|^2 = 1 \tag{7}$$

so the determinant of the rotation matrix can be either ± 1,

$$\left|\underline{\underline{R}}\right| = \pm 1; \tag{8}$$

8. Continuous motions of a rigid body must preserve the value of the determinant of the rotation matrix, which is +1 to preserve right-handedness of the coordinate systems.

Each column of a rotation matrix represents the components of a vector of length 1 and each row of a transposed rotation matrix represents the components of a vector of length 1. The three column vectors in $\underline{\underline{R}}$ are the unit vectors $\mathbf{e}'_i$ into which $\mathbf{e}_i$ are mapped by $\underline{\underline{R}}$, that is,

$$\underline{\underline{R}} = \left|\mathbf{e}'_1(\mathbf{e}_i) \quad \mathbf{e}'_2(\mathbf{e}_i) \quad \mathbf{e}'_3(\mathbf{e}_i)\right|. \tag{9}$$

The column vectors must be mutually orthogonal. Since $\underline{\underline{R}}\underline{\underline{R}}^T = 1$ these facts represent *six* conditions that must be imposed upon the elements of a rotation matrix. Thus, an orthogonal (rotation) matrix represents *three* degrees of freedom of the rigid body. The total number of degrees of freedom of a rigid body include three for translation plus the three for rotation as stated above, giving a total of six degrees of freedom.

Every rotation matrix has an eigenvector which it preserves, i. e.,

$$\underline{\underline{R}}\mathbf{x}_R = \mathbf{x}_R \tag{10}$$

where $\mathbf{x}_R$, the eigenvector, represents the *axis* of the rotation described by $\underline{\underline{R}}$. This equation can be written as

$$\left(\underline{\underline{R}}-\lambda\underline{\underline{I}}\right)\mathbf{x}_R=0 \tag{11}$$

($\underline{\underline{I}}$ is the identity matrix) which can have a solution only if the determinant is zero,

$$\left|\underline{\underline{R}}-\lambda\underline{\underline{I}}\right|=0. \tag{12}$$

This cubic equation in λ has three roots, one of which must be equal to 1. The absolute value to the root is 1 and the product of all three roots is also 1. If all three roots are 1, the rotation is the null rotation. In all possible cases, $\mathbf{x}_R$, the axis of the rotation exists.

For an infinitesimal rotation

$$\underline{\underline{R}}=\underline{\underline{I}}+d\alpha_i\underline{\underline{L}}_i \tag{13}$$

where $d\alpha_i$ represent the components of the infinitesimal rotation vector having three independent infinitesimal angles $d\alpha_i$ and

$$\underline{\underline{L}}_1=\begin{vmatrix}0&0&0\\0&0&-1\\0&1&0\end{vmatrix};\quad \underline{\underline{L}}_2=\begin{vmatrix}0&0&1\\0&0&-0\\-1&0&0\end{vmatrix};\quad \underline{\underline{L}}_3=\begin{vmatrix}0&-1&0\\1&0&-1\\0&0&0\end{vmatrix}. \tag{14}$$

$\mathbf{d}\alpha$ can be considered to be a vector only for infinitesimal rotations. It actually represents a pseudo-vector which is reversed by a mirror reflection (its components are identified with the three components of a skew-symmetric rotation matrix–see Katz (1997)). The axis of rotation can be represented by a unit vector in the direction of $\mathbf{d}\alpha$, i. e., $\mathbf{d}\alpha=\mathbf{e}\,d\alpha$. The effect of $\underline{\underline{R}}$ given above on a vector $\mathbf{v}$ is to change it into

$$\mathbf{v}-d\alpha(\mathbf{e}\times\mathbf{v}) \tag{15}$$

so that the differential equation for the effect of the rotation on the vector is

$$\frac{d\mathbf{v}}{d\alpha}=(\mathbf{e}\times\mathbf{v}). \tag{16}$$

Since the angular velocity is $d\alpha/dt=\omega$ this equation can be written as

$$\frac{d\mathbf{v}}{dt}=(\omega\times\mathbf{v}). \tag{17}$$

Now the above results must be repeated for finite rotations. For finite rotations the rotation matrix can be expanded as

$$\underline{\underline{R}} = \underline{\underline{I}} + d\alpha \mathbf{e} \cdot \underline{\underline{L}}\mathbf{x}. \tag{18}$$

Operating $\underline{\underline{R}}$ on a position vector and differentiating the final expression leads to the differential equation for the change of the position vector with time

$$\frac{d\mathbf{x}}{dt} = \mathbf{e} \cdot \underline{\underline{L}}\mathbf{x}. \tag{19}$$

This equation can be integrated and expanded to give the following expression for the rotation matrix

$$\underline{\underline{R}}(\mathbf{e}, \alpha) = \underline{\underline{I}} \cos\alpha + (1 - \cos\alpha)\mathbf{e} \cdot \mathbf{e}^{\mathrm{T}} + \underline{\underline{L}}\mathbf{e} \sin\alpha \tag{20}$$

or in component form

$$\underline{\underline{R}}_{ij} = \delta_{ij} \cos\alpha + e_i e_j (1 - \cos\alpha) - \varepsilon_{ijk} e_k \sin\alpha. \tag{21}$$

In this expression δ_{ij} is the Kronecker delta which is equal to 1 if i = j and 0 otherwise and ε_{ijk} is the permutation symbol having the following values

$$\varepsilon_{ijk} = \begin{cases} 0, \textit{ if any two of } i, j, k \textit{ are the same} \\ 1, \textit{ if } ijk \textit{ is an even permutation of } 1,2,3. \\ -1, \textit{ if } ijk \textit{ is an odd permutation of } 1,2,3 \end{cases} \tag{22}$$

Using this expansion of the rotation matrix, the angle of rotation can be found from the trace of the matrix, that is,

$$\cos\alpha = \frac{1}{2}\left(\underline{\underline{R}}_{ii} - 1\right), \tag{23}$$

and the components of the unit vector in the direction of the rotation (using the right hand rule for connecting the direction and sense of rotation) from the following relations

$$e_1 = \frac{\underline{\underline{R}}_{32} - \underline{\underline{R}}_{23}}{2\sin\alpha}, \tag{24a}$$

$$e_2 = \frac{\underline{\underline{R}}_{13} - \underline{\underline{R}}_{31}}{2\sin\alpha}, \tag{24b}$$

$$e_3 = \frac{\underline{\underline{R}}_{21} - \underline{\underline{R}}_{12}}{2\sin\alpha}. \tag{24c}$$

Euler's theorem states that the most general continuously attainable rigid body motion in space can be represented as a continuous rotation around a fixed axis (plus translation of the center of mass). That is, any orientation of a rigid body, however actually reached, can be reached through a single rotation around a fixed axis **e** by an angle α which is in the range $0 \le \alpha \le \pi$. Successive rotations described by the rotation matrices $\underline{\underline{R}}^1$, followed by $\underline{\underline{R}}^2$, are equivalent to a single rotation described by the rotation matrix

$$\underline{\underline{R}}' = \underline{\underline{R}}^2 \underline{\underline{R}}^1. \tag{25}$$

For a continuous rotation the rate of change of the rotation matrix is given by the differential equation

$$\frac{d\underline{\underline{R}}}{dt} = \underline{\underline{\Omega R}} \tag{26}$$

where matrix multiplication is indicated and the matrix $\underline{\underline{\Omega}}$ is related to the angular velocity vector ω by the following skew symmetric relation

$$\underline{\underline{\Omega}} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{27}$$

This expression assumes that we are considering *active* rotations where the body is continuously moved from one orientation to another as viewed in the earth-axis coordinate system.

Quaternions

Rigid body rotations are normally described in aeronautics by the Euler angles. However, they have a singularity at orientations which represent motion toward the poles in spherical geometry. Quaternions describe all orientations without singularity and are thus more useful for aerodynamic vehicles that undergo extremes of changes of orientation. In addition, they describe rotational changes with four parameters with one

constraint, while the rotation matrices, which also describe all orientation changes without singularity, require six parameters and three constraints.

Quaternions use four units: 1, $\mathbf{i}, \mathbf{j}, \mathbf{k}$ which obey the following multiplication rules

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1 \tag{28}$$

$$\begin{aligned} \mathbf{ij} &= -\mathbf{ji} = \mathbf{k} \\ \mathbf{jk} &= -\mathbf{kj} = \mathbf{i} \\ \mathbf{ki} &= -\mathbf{ik} = \mathbf{j}. \end{aligned} \tag{29}$$

An arbitrary quaternion can be written in the form

$$Q = Q_o + Q_x\mathbf{i} + Q_y\mathbf{j} + Q_z\mathbf{k} \tag{30}$$

or

$$Q = Q_o + \mathbf{Q} \tag{31}$$

where $\{Q_o, Q_x, Q_y, Q_z\}$ are the *components* of the quaternion. A quaternion can be thought to consist of a scalar part, Q_o, and a vector part, $\mathbf{Q}$. The vectors $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ can be thought of as being unit vectors in the three coordinate directions. The quaternions can also be represented by the following 4×4 matrices:

$$\mathbf{1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{32}$$

$$\mathbf{i} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{33}$$

$$\mathbf{j} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{34}$$

$$\mathbf{k} = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{35}$$

Quaternion addition is commutative and associative and multiplication is associative and distributive over addition. However, quaternion multiplication is not commutative. If A, B, C are quaternions the following operations are valid:

$$\begin{aligned} &A+B=B+A; \\ &A+B+C=(A+B)+C=A+(B+C); \\ &(AB)C=A(BC); \\ &A(B+C)=AB+AC; \\ &(A+B)C=AC+BC. \end{aligned} \tag{36}$$

However,

$$AB \neq BA. \tag{37}$$

Using the mathematical operations above plus the rules for multiplying the unit vectors, the product of two quaternions, Q and P can be written in terms of usual mathematical operations as

$$\begin{aligned} (Q_o+\mathbf{Q})(P_o+\mathbf{P}) = Q_oP_o - \mathbf{Q}\cdot\mathbf{P} + \\ Q_o\mathbf{P} + P_o\mathbf{Q} + \mathbf{Q}\times\mathbf{P}. \end{aligned} \tag{38}$$

This form, which contains the vector scalar and vector products, contains a scalar part (first two terms) and a vector part.

The conjugate of a quaternion Q is

$$(Q_o+\mathbf{Q})^* = Q_o - \mathbf{Q}. \tag{39}$$

The product of a quaternion and its conjugate is a semidefinite number

$$QQ^* = Q_o^2 + \mathbf{Q}^2 \tag{40}$$

and the absolute value of a quaternion is defined as

$$|Q| = (QQ^*)^{1/2}. \tag{41}$$

The absolute value of the product of quaternions is the product of the absolute values of the individual quaternions:

$$|PQR...| = |P||Q||R|... \tag{42}$$

On the other hand, the conjugate of the product of quaternions is the inverse product of the conjugates of the individual quaternions:

$$(PQR....)^* = ...R^*Q^*P^*. \tag{43}$$

A quaternion $Q = 0$ only if $Q_o = Q_x = Q_y = Q_z = 0$, that is, only if all of its components are zero. Then the product of quaternions cannot be zero unless at least one of the individual quaternions is zero. The inverse of a quaternion that is non-zero is easily computed by the equation

$$Q^{-1} = \frac{Q^*}{|Q|^2}. \tag{44}$$

The inverse is such that it commutes with itself, the product being unity.

$$QQ^{-1} = Q^{-1}Q = 1. \tag{45}$$

The rotation matrix for the space rotation of a three-dimensional body is described by a *unit quaternion*, that is, one whose absolute value is unity. For a unit quaternion, designated by a lower case q, the inverse is equal to the transpose, i. e.,

$$q^* = q^{-1}. \tag{46}$$

All unit quaternions can be written in terms of a unit vector $\mathbf{e}$, which is in the direction of rotation, and an angle β, which is half of the angle of rotation, by the equation

$$q = \cos\beta + \mathbf{e}\sin\beta. \tag{47}$$

A vector $\mathbf{x}$ is transformed into the vector $\mathbf{x}'$ by an active rotation. This process can be described by the following unit quaternion multiplication

$$\mathbf{x}' = q\mathbf{x}q^*. \tag{48}$$

This operation is equivalent to the operation of multiplying the vector $\mathbf{x}$ by the rotation matrix to get the rotated vector $\mathbf{x}'$. The operation is linear and represents a rotation about the origin, as previously discussed. Using the above form for q, the transformed vector can be written in the form

$$\mathbf{x}' = \cos(2\beta)\mathbf{x} + \{1 - \cos(2\beta)\}(\mathbf{e}\cdot\mathbf{x})\mathbf{e} + \sin(2\beta)(\mathbf{e}\times\mathbf{x}). \tag{49}$$

This is identical to the result previously obtained in Eq. (20) for the rotation of the vector $\mathbf{x}$ by an angle 2β around the axis designated by the unit vector $\mathbf{e}$. Thus, the equations equivalent to those that use the rotation matrix are

$$\mathbf{x}' = q\mathbf{x}q^* \tag{50}$$

and

$$q = \cos\left(\frac{\alpha}{2}\right) + \mathbf{e}\sin\left(\frac{\alpha}{2}\right). \tag{51}$$

Now consider the rotation of α about the axis described by the unit vector $\mathbf{e}$, followed by a rotation by an angle α' about the axis described by the unit vector $\mathbf{e}'$. If the first rotation is described by the quaternion q and the second by the quaternion q', then the resultant rotation is described by the product qq', i. e., $\mathbf{x} \rightarrow \mathbf{x}' \rightarrow \mathbf{x}''$ is given by

$$\mathbf{x}'' = (q'q)\mathbf{x}(q'q)^*. \tag{52}$$

Thus, the rule for combining rotations is that of multiplication of quaternions. Use of this form simplifies the computation of the resultant rotation axis and angle.

Now reconsider an infinitesimal rotation. It may be represented by the quaternion

$$dq = 1 + \frac{1}{2}\omega dt \tag{53}$$

or

$$\frac{dq}{dt} = \frac{1}{2}\omega q \tag{54}$$

where ω is the angular velocity and a quaternion product is indicated in the last equation. This last equation is equivalent to four individual equations, one equation for each component of the quaternion. It is equivalent to the differential equation for the rate of change of the rotation matrix given in Eq. (26). Both equations are linear, whereas the Euler equations will be seen to be nonlinear. Notice that Eq. (26) represents nine differential equations for the components of the rotation matrix. Writing the quaternion as $q = q_o + \mathbf{q}$, we can split Eq. (54) into a scalar and a vector equation, as follows:

$$\frac{dq_o}{dt} = \frac{1}{2}\omega \cdot \mathbf{q},$$
$$\frac{d\mathbf{q}}{dt} = \frac{1}{2}\omega q_o + \frac{1}{2}\omega \times \mathbf{q}. \tag{55}$$

Once the quaternion is known, the rotation can be determined by quaternion multiplication. The rotation matrix can also be represented in term of the quaternion components.

$$\underline{\underline{R}} = \begin{pmatrix} q_o^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y - 2q_zq_o & 2q_xq_z + 2q_yq_o \\ 2q_yq_x + 2q_zq_o & q_o^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z - 2q_xq_o \\ 2q_zq_x - 2q_yq_o & 2q_zq_y + 2q_xq_o & q_o^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix} \tag{56}$$

The transpose of the rotation matrix is given by

$$\underline{\underline{R}}^T = \begin{pmatrix} q_o^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y - 2q_zq_o & 2q_xq_z + 2q_yq_o \\ 2q_xq_y - 2q_zq_o & q_o^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z - 2q_xq_o \\ 2q_xq_z + 2q_yq_o & 2q_zq_y + 2q_xq_o & q_o^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix} \tag{57}$$

The quaternion for the null rotation of $\alpha = 0$ is the number 1 while the quaternion for the equivalent rotation of $\alpha = 2\pi$ is -1. Thus the rotation induced by $-q$ is the same as that induced by q.

The absolute value of the quaternion must remain equal to one during the motion of the body. This is the only constraint that must be imposed on the quaternion. Numerical error will allow the magnitude of the quaternion to drift from one and it must be renormalized to maintain the constraint, normally at each iteration step. Various schemes have been developed for normalization. Katz (1997)) multiplied each component by

$$|q|^{-1} \approx \frac{3}{2} - \frac{1}{2}q^*q \tag{58}$$

at each iteration step whereas Gebert, et. al (1998) added diagonal elements $k\varepsilon$ to the diagonal to Eq. (54) when it is written in matrix form, where

$$\varepsilon = 1 - \left(q_o^2 + q_x^2 + q_y^2 + q_z^2\right) \tag{59}$$

and k is a constant set equal to 0.1.

Before continuing it must be pointed out the Gebert, et. al (1998) followed Robinson (1958) and defined the terms of the quaternion differently from the discussion given to this point. For these authors the quaternion components are given the symbols e_i, $i = 1, 2, 3, 4$ and the quaternion is written as

$$q = e_1 + e_4\mathbf{i} + e_3\mathbf{j} + e_2\mathbf{k} \tag{60}$$

where the usual unit vectors in the x-, y-, z-directions are the same as before. This renumbering of the quaternion elements can lead to confusion when attempting to compare various works that use the quaternion approach. With this nomenclature the quaternion update equation used by Gebert, et. al (1998) is

$$\begin{vmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \\ \dot{e}_4 \end{vmatrix} = \begin{vmatrix} k\varepsilon & -\frac{1}{2}\omega_z & -\frac{1}{2}\omega_y & -\frac{1}{2}\omega_x \\ \frac{1}{2}\omega_z & k\varepsilon & -\frac{1}{2}\omega_x & \frac{1}{2}\omega_y \\ \frac{1}{2}\omega_y & \frac{1}{2}\omega_x & k\varepsilon & -\frac{1}{2}\omega_z \\ \frac{1}{2}\omega_x & -\frac{1}{2}\omega_y & \frac{1}{2}\omega_z & k\varepsilon \end{vmatrix} \begin{vmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{vmatrix}. \tag{61}$$

Active and Passive Rotations of Vectors and Matrices

To this point the active rotation of a position vector fixed on a rigid body has been discussed. An *active* rotation was defined as the motion of a body as viewed from a fixed coordinate system (e. g., the earth-based system) so that the rotation maps each point in the body to a new corresponding point. Most texts, on the other hand, describe *passive* rotations where the coordinate system attached to the body rotates. It is always assumed that all coordinate systems are right handed.

In the case of a passive rotation, we consider a coordinate system (x, y, z) attached the body and a second system (x', y', z') rotated with respect to the first. The rotation of the second system relative to the first is described by a rotation matrix. The coordinate system (x, y, z) defines the position relative to the axes of the coordinate system.

Applying a rotation to the coordinate system has the same effect as applying the opposite rotation to the point. If the rotation is defined by the rotation matrix $\underline{\underline{R}}$, then the passive rotation is described by the transposed rotation matrix, $\underline{\underline{R}}^T$. If an active rotation is described by the quaternion q the passive rotation is described by q^* (since q is a unit quaternion, $q^T = q^*$). The report by Gebert, et. al (1998) used passive rotations rather than active.

Thus far we have only discussed the rotation of position vectors attached to a rigid body as it rotates in space. However, the same rotation matrix can be used to describe the rotation of velocity, linear acceleration, angular velocity, eand other vectors. Scalars are unaffected by rotation.

The rotation matrix can also be used to describe the rotation of tensor quantities. Consider a matrix which operates on a column vector $\mathbf{v}$ to generate a column vector $\mathbf{u}$, i. e.,

$$\mathbf{u} = \underline{\underline{A}}\mathbf{v}. \tag{62}$$

We want to see how the matrix $\underline{\underline{A}}$ transforms due to an active rotation of the body.. As a result of an active rotation $\mathbf{v}$ is transformed into $\mathbf{v}'$ by the relation

$$\mathbf{v}' = \underline{\underline{R}}\mathbf{v}. \tag{63}$$

Then it can be shown that

$$\mathbf{u}' = \underline{\underline{A}}'\mathbf{v}' \tag{64}$$

where the transformation matrix is determined by the relation

$$\underline{\underline{A}}' = \underline{\underline{R}}\,\underline{\underline{A}}\,\underline{\underline{R}}^T. \tag{65}$$

Eq. (65) gives the rule for the active rotation of matrices, the most common of which is the moment of inertia matrix but includes such things as the stress and strain tensors and the rotation matrix itself.

$\underline{\underline{R}}$ describes the active rotation of a rigid body which has a particular system of coordinates attached to it. Suppose a different set of body coordinates is used. If the passive rotation matrix between the first coordinate system and the new set is $\underline{\underline{\tilde{R}}}$, then the rotation matrix in the new coordinate system is (passive rotation)

$$\underline{\underline{R'}} = \underline{\underline{\tilde{R}}}^T \underline{\underline{R}}\underline{\underline{\tilde{R}}}. \tag{66}$$

All of the quantities in this relation are rotation matrices. A similar relation applies for the transformation of quaternions due to the passive rotation to a new coordinate system,

$$q' = p^* q p, \tag{67}$$

where quaternion multiplication is indicated. If q is a unit quaternion then q' is the quaternion describing the same rotation of the body after the coordinate system has been rotated by the quaternion p.

Euler Angles

The Euler angles are a common three-parameter method for describing the rotation of a rigid body in space. Various authors use different sets of Euler angles (compare Goldstein (1950) and Katz (1997)) and this can lead to great confusion. Some coordinate transformations are even left handed. As previously discussed, rigid body rotation in space can be described by a minimum of four parameters and so it is anticipated the the Euler angles cannot adequately describe all of three-eimensional rotational space.

The method used by Katz (1997) for describing the Euler angles, which is probably the one most commonly used in aeronautics, will be described here. First, the normal body-axes coordinate system must be described. Assume that the flight vehicle has longitudinal symmetry about its mid plane. Then the x^b coordinate will point "forward", the z^b axis will point "downward" and the y^b will complete the right-handed coordinate system, pointing along the right wing of an airplane.

Now consider the definition of the Euler angles given by Katz (1997). Start with a reference orientation, with the wings level for an airplane, for example. First yaw the vehicle by the angle ψ until the final heading is reached $(-\pi \le \psi \le \pi)$. Then pitch the vehicle by the angle θ until the final pitch is reached $\left(-\frac{1}{2}\pi \le \theta \le \frac{1}{2}\pi\right)$ and finally roll the vehicle by the angle φ until the final roll is reached $(-\pi \le \varphi \le \pi)$. These are active rotations of the vehicle which are defined relative to the body axes as altered by the previous rotation(s).

The rotations can be defined in terms of matrices that are written in the body axes as follows:

$$\begin{aligned}
\underline{\underline{\Psi}} &= \begin{vmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{vmatrix}; \\
\underline{\underline{\Theta}} &= \begin{vmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{vmatrix}; \\
\underline{\underline{\Phi}} &= \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{vmatrix}.
\end{aligned} \tag{68}$$

Since there matrices are defined in body axes, the resulting rotation matrix is given by

$$\underline{\underline{R}} = \underline{\underline{\Psi\Theta\Phi}}. \tag{69}$$

The same result could be obtained in reverse order by performing the rotations using the earth-based coordinate system but the process described is the one usually given in the text books.

The rotation matrix that results from multiplying the three matrices in Eq. (69) together is

$$\underline{\underline{R}} = \begin{vmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\varphi - \sin\psi\cos\varphi & \cos\psi\sin\theta\cos\varphi + \sin\psi\sin\varphi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\varphi + \cos\psi\cos\varphi & \sin\psi\sin\theta\cos\varphi - \cos\psi\sin\varphi \\ -\sin\theta & \cos\theta\sin\varphi & \cos\theta\cos\varphi \end{vmatrix}. \quad (70)$$

This is the matrix that represents the active rotation from the reference orientation to the current orientation. Recall that the maneuvers which the vehicle undertook to achieve its final orientation do not enter into the rotation matrix. The rotation matrix describes the result of the individual rotations that were required to achieve the current orientation.

Every set of Euler angles defines a rotation matrix. However, not all sets of angles define different physical orientations of the vehicle, since many combinations of φ and ψ give the same rotation matrix. Ti will also be seen that there is a singularity in the kinematical equations when $\theta = \pm\pi/2$. For details see Katz (1997). For these reasons the Euler angles are not useful for use with Virtual Flight Testing.

The Euler angles can be translated into quaternions, one quaternion defined for each separate rotation. These quaternions are

$$\begin{aligned} \Psi &= \cos\frac{\psi}{2} + \mathbf{k}\sin\frac{\psi}{2}; \\ \Theta &= \cos\frac{\theta}{2} + \mathbf{j}\sin\frac{\theta}{2}; \\ \Phi &= \cos\frac{\varphi}{2} + \mathbf{i}\sin\frac{\varphi}{2}. \end{aligned} \quad (71)$$

For active rotation the quaternion for the final orientation is

$$q = \Psi\Theta\Phi. \quad (72)$$

The time derivatives of the Euler angles can be shown to be (Katz (1997))

$$\begin{aligned} \dot{\psi} &= \omega_z - \left(\omega_x\cos\psi + \omega_y\sin\psi\right)\tan\theta; \\ \dot{\theta} &= \omega_y\cos\psi - \omega_z\sin\psi; \\ \dot{\varphi} &= \frac{\omega_x\cos\psi + \omega_y\sin\psi}{\cos\theta}. \end{aligned} \quad (73)$$

where the components of ω are defined in the earth axes. These equations are equivalent to Eq. (26) for the rotation matrix and Eq. (54) for the quaternions, all for active rotations defined in earth axes. The singularity for $\theta = \pm \pi/2$ is evident from the last equation. Also notice that they involve the evaluation of many trigonometric functions which is time consuming on the computer.

Derivation of VFT Dynamical Equations

In this final section the derivation of the dynamical simulation equations as given in Gebert, et. al (1998) will be outlined. They introduced a very complex notation that will not be used here. In addition, only the equations for the unconstrained flight vehicle in free space will be given since it illustrates the methods used in the report. In general, passive rotations are used throughout the report, using the quaternion notation given by Eq. (60). This notation obviously came from early use of the Euler angles since the Euler angles are produced by performing rotations around the coordinate axes taken in an inverse order (rotation about the z-axis, then the y and finally the x) (see Eq. (71)).

First we must consider the equations for the dynamics of a rigid body in space. The linear momentum is given by (see Goldstein (1950))

$$\mathbf{P} = m\frac{d\mathbf{X}}{dt} \tag{74}$$

where m is the total mass of the body, $\mathbf{X}$ is the location of its center of mass and $\mathbf{P}$ is its total linear momentum. The center of mass of the rigid body is chosen as the point to attach the body-fixed coordinate system. Then Newton's second law becomes

$$\frac{d\mathbf{P}}{dt} = \mathbf{F}, \tag{75}$$

where $\mathbf{F}$ is the resultant of all forces that act on the body.

For simplicity the rigid body can be divided into a number of mass particles. Define a system of coordinates for the i^{th} mass particle in the earth-based system as follows:

$$\mathbf{x}^i = \mathbf{X} + \mathbf{r}^i. \tag{76}$$

Since $\mathbf{r}^i$, the position of the particle with respect to the center of mass, is defined in the earth-based coordinate system, it does not rotate when the body changes angular orientation. Then the angular momentum of the system of mass particles becomes

$$\mathbf{K} = \sum_i m^i \mathbf{r}^i \times \mathbf{r}^i \tag{77}$$

and the rate of change of the system angular momentum is

$$\frac{d\mathbf{K}}{dt} = \mathbf{M} = \sum_i \mathbf{r}^i \times \mathbf{F}^i \tag{78}$$

where $\mathbf{F}^i$ is the resultant force acting upon the i^{th} mass particle.

The following equations apply for a rigid body where the motion of the mass particles is related to one another through the angular velocity of the body. The angular momentum can be written as

$$\begin{aligned} &\dot{\mathbf{r}}^i = \omega \times \mathbf{r}^i; \\ &\mathbf{K} = \sum_i m^i \left((r^i)^2 \omega - (\mathbf{r}^i \cdot \omega) \mathbf{r}^i \right); \\ &\mathbf{K} = \underline{\underline{J}} \omega; \end{aligned} \tag{79}$$

where $\underline{\underline{J}}$ is the moment of inertia tensor which is characteristic of the mass distribution of the rigid body. The moment of inertia tensor is determined by the following integrals over the mass distribution of the body:

$$\begin{aligned} &\underline{\underline{I}}_{ij} = \int x_i x_j dm, \; i \neq j; \\ &\underline{\underline{I}}_{11} = \int (x_2^2 + x_3^2) dm, \; etc. \end{aligned} \tag{80}$$

Then the moment of inertia tensor is defined in terms of these integrals as follows:

$$\underline{\underline{J}} = \begin{vmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{vmatrix}. \tag{81}$$

$\underline{\underline{J}}$ is defined in earth coordinates and varies as the body rotates according to the equation

$$\underline{\underline{J}}^e = \underline{\underline{R}}\,\underline{\underline{J}}^b\,\underline{\underline{R}}^T. \tag{80}$$

where the superscripts e and b designate the earth and body coordinate systems, respectively.

The basic dynamical equations are Eqs. (78) and (79) where $\mathbf{K}$, $\mathbf{M}$, ω, and $\underline{\underline{J}}$ are defined in earth coordinates. Putting the equation together we get

$$\frac{d}{dt}\left(\underline{\underline{J}}^e\omega^e\right) = \mathbf{M}^e. \tag{81}$$

Since all quantities are defined in earth coordinates, they all vary with time. We must transform them to body coordinates to make the moment of inertia tensor constant. This is done by operating the rotation matrix on all of them, as in Eq. (63). The equations are

$$\begin{aligned} \omega^b &= \underline{\underline{R}}^T\omega^e; \\ \mathbf{K}^b &= \underline{\underline{R}}^T\mathbf{K}^e; \\ \mathbf{M}^b &= \underline{\underline{R}}^T\mathbf{M}^e, \end{aligned} \tag{82}$$

and Eq. (80) for $\underline{\underline{J}}$. In these equations $\underline{\underline{R}}$ is the rotation matrix for the active rotation from the earth to the body coordinates. In body coordinates it varies with time as

$$\frac{d\underline{\underline{R}}}{dt} = \underline{\underline{R}}\,\underline{\underline{\Omega}} \tag{83}$$

with $\underline{\underline{\Omega}}$ given by Eq. (27) where the components of ω are defined in the body-axis system. Putting these equations together, and using the fact the $\underline{\underline{R}}^T\underline{\underline{R}} = 1$, gives the equation

$$\frac{d}{dt}\left(\underline{\underline{R}}\,\underline{\underline{J}}^b\omega^b\right) = \underline{\underline{R}}\mathbf{M}^b. \tag{84}$$

Using the fact that $\underline{\underline{J}}^b$ is a constant plus Eq. (83), we get the final result

$$\underline{\underline{J}}^b\frac{d\omega^b}{dt} + \underline{\underline{\Omega}}\,\underline{\underline{J}}^b\omega^b = \mathbf{M}^b. \tag{85}$$

This last Eq. (85) are the familiar Euler's equations, which are not simple unless the body has enough symmetry for the moment of inertia tensor to be diagonal for some body coordinate system. When expanded this equation is Eq. (7.2) in Gebert, et. al (1998),

where they have solved the equation for the rates of change of the components of angular velocity:

$$\begin{vmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{vmatrix} = \begin{vmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{vmatrix}^{-1} \times$$
$$\left\{ \begin{vmatrix} M_x + I_{xz}\omega_x\omega_y + I_{yz}\left(\omega_y^2 - \omega_z^2\right) \\ M_y + I_{yz}\omega_x\omega_y + I_{xz}\left(\omega_x^2 - \omega_z^2\right) \\ M_z - I_{xz}\omega_x\omega_z + I_{xy}\left(\omega_y^2 - \omega_x^2\right) \end{vmatrix} \right\} + \tag{86}$$
$$\begin{vmatrix} \left(I_{yy} - I_{zz}\right)\omega_y\omega_z - I_{xy}\omega_x\omega_z \\ \left(I_{xx} - I_{zz}\right)\omega_x\omega_z - I_{xy}\omega_y\omega_z \\ \left(I_{yy} - I_{xx}\right)\omega_x\omega_y + I_{xz}\omega_y\omega_z \end{vmatrix}.$$

Finally, to conclude the derivation of the dynamical equations of motion for a free body, the gravitational force vector must be transformed between the two coordinate systems. Gebert, et al (1998) used a passive rotation so that the g-vector is transformed using the equation

$$\mathbf{g}^b = q^* \mathbf{g} q \tag{87}$$

where the quaternion of the transformation is given in Eq. (60). Written out this becomes

$$\mathbf{g}^b = g\left[2\left(e_2e_4 - e_1e_3\right)\mathbf{i} + 2\left(e_1e_4 - e_2e_3\right)\mathbf{j} + 2\left(e_1^2 + e_2^2 - e_3^2 - e_4^2\right)\mathbf{k}\right]. \tag{88}$$

Using this form, the momentum equation becomes

$$\begin{Bmatrix} \dot{V}_x \\ \dot{V}_y \\ \dot{V}_z \end{Bmatrix} = \begin{Bmatrix} \frac{F_x}{m} + 2g\left(e_2e_4 - e_1e_3\right) - \omega_y V_z + \omega_z V_y \\ \frac{F_y}{m} + 2g\left(e_2e_3 - e_1e_4\right) - \omega_z V_x + \omega_x V_z \\ \frac{F_z}{m} + 2g\left(e_1^2 + e_2^2 - e_3^2 - e_4^2\right) - \omega_x V_y + \omega_y V_x \end{Bmatrix}. \tag{89}$$

This is the form used in Eq. (7.1) in Gebert, et. al (1998).

Concluding Remarks

This report has covered one step in the process of developing the Virtual Flight Testing concept into a usable method for testing some of the dynamic aspects of a flight vehicle in a wind tunnel. VFT will require real-time computation of the kinematics of the model of the vehicle while it is undergoing rapid orientational changes in the wind tunnel. It is concluded that the quaternion approach is superior to the Euler angle approach because it can be used for all angular orientations without singularity and does not require the computationally-intensive evaluation of trigonometric functions. It is also computationally superior to the rotation matrix formulation since it involves fewer equations and only one constraint must be imposed, compared to six for the rotation matrix. The next steps for VFT are under active development at the Arnold Engineering Development Center with assistance from personnel at Eglin Air Force Base and China Lake Naval Air Weapons Center.

References

1. Jack C. K. Chou, "Quaternion Kinematic and Dynamic Differential Equations," IEEE Trans. Robotics and Automation, Vol. 8, No. 1, Feb. 1992, pp 53-64.
2. Glenn Gebert, Joy Kelly and Juan Lopez, "Virtual Flight Test VFT) Modeling and Assessment," 30 September 1998.
3. Herbert Goldstein, *Classical Mechanics*, Addison-Wesley, Cambridge, Mass., 1950.
4. Amnon Katz, *Computational Rigid Vehicle Dynamics*, Krieger Publishing Co., Malabar, Florida, 1997.
5. Jack B. Kuipers, *Quaternions and Rotation Sequences*, Princeton University Press, Princeton, New Jersey, 1999.
6. Alfred C. Robinson, "On The Use of Quaternions in Simulation of Rigid-Body Motion," Wright Air Development Center Technical; Report 58-17, December 1958.

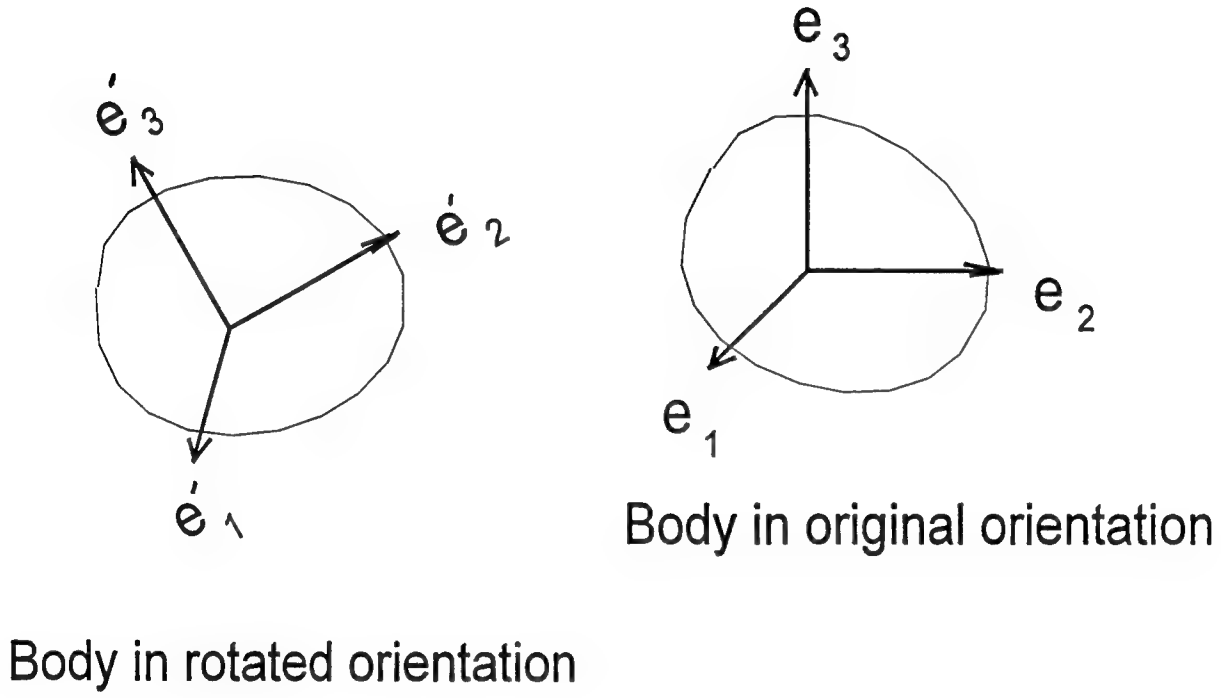


Figure 1. Initial and Final Body Locations.

STATIC STRENGTH PROPERTIES OF NATURALLY CORRODED AND UNCORRODED AGED AIRCRAFT ALUMINUM SKIN MATERIALS

P. W. Whaley
Professor
Department of Mechanical Engineering

and

S. N. Gurney
Undergraduate Student
Department of Mechanical Engineering

Oklahoma Christian University
2501 East Memorial Road
Oklahoma City, OK 73136

Final Report for:
Summer Research Extension Program
Oklahoma City Air Logistics Center

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

September 1998

STATIC STRENGTH PROPERTIES OF NATURALLY CORRODED AND UNCORRODED AGED AIRCRAFT ALUMINUM SKIN MATERIALS

P. W. Whaley
Professor

and

S. N. Gurney
Undergraduate Student
Department of Mechanical Engineering
Oklahoma Christian University

Abstract

The yield and ultimate strengths of naturally corroded and uncorroded aged 2024T3 Aluminum aircraft skin materials were compared with the new alloy. Skin materials were obtained from aircraft undergoing Programmed Depot Maintenance at the Oklahoma City Air Logistics Center. Results show that there is no reduction in strength due to aging without corrosion. The effect of natural corrosion was quantified by comparing the strength of corroded specimens with uncorroded specimens taken from the same panel. Corrosion damage was described by the qualitative terms: very light, light, moderate and severe. Results show that there is more variation of strength in the corroded specimens than in the uncorroded specimens. That is not surprising since equivalent material thickness loss is very difficult to measure in naturally corroded specimens due to non uniform nature of the corrosion. The qualitative description of corrosion damage is inadequate since it introduces uncertainty in the amount of thickness loss. For severely corroded specimens there was a significant reduction in the strain at fracture as well as ultimate strength. Future research should focus on providing a corrosion damage measure that quantifies the severity of corrosion damage as correlated with structural strength measurements.

STATIC STRENGTH PROPERTIES OF NATURALLY CORRODED AND UNCORRODED AGED AIRCRAFT ALUMINUM SKIN MATERIALS

P. W. Whaley and S. N. Gurney

Introduction

The Air Force operates a large fleet of C/KC-135 aircraft, many of them more than 40 years old. Decreasing defense budgets make it unlikely that this fleet will be replaced for several more years. This puts a tremendous burden on Programmed Depot Maintenance (PDM), since the Air Force's need for these aircraft is not diminished. The C/KC-135 aircraft is well designed and fatigue cycles are accumulating slowly enough at the current usage to allow these aircraft to remain in service for several years. However, spot-welded fabrication techniques used in the 1950's in an aircraft designed to operate for 20 years are now causing corrosion damage. The spot-welded skin doublers used for extra strength in critical locations were assembled without corrosion protective coatings because of the need for an electrical conduction path. These locations of bare metal contact with moisture and other contaminates are now sites for corrosion. Many of these fuselage skins are being replaced during PDM.

The influence of corrosion damage on structural reliability is not well understood. At present, corrosion is described by the qualitative terms: light, moderate and severe. A universal definition of these terms does not exist. The absence of a quantitative measure of the extent of corrosion makes the task of identifying and repairing corrosion damage much more difficult. Nondestructive inspection procedures are being developed to permit early detection of corrosion, but these techniques are not likely to be in reliable, widespread usage soon. It is possible that some corrosion damage will be missed in some aircraft leaving PDM. Therefore, it is very important to develop techniques for evaluating the influence of undetected corrosion.

Corrosion damage in the skin doublers is a random, non-uniform phenomenon. Spots of various levels of corrosion can be found throughout the skin doublers between the rivet patterns. Skin doublers that were cut between the spot welds near areas of severe corrosion show no corrosion a short distance away. Inside every skin doubler spots of corrosion always appear between areas of apparently uncorroded material. In all of the skins examined, corrosion by-products are found between the skin doublers. The only uncorroded skin material was a single layer.

In the absence of an accurate, quantitative, engineering description of the effect of corrosion, many engineers use the equivalent material thickness loss as a tool in the evaluation of corrosion

damage. There is compelling evidence that this approach may be valid when predicting crack growth in the aluminum alloys 2024T3, 2024T4, 7075T6 and 7178T6 (Luzar, 1997). When the historical scatter in crack growth rate for those alloys are included in the data, corroded material crack growth rate falls within the uncorroded scatter bands.

Measurement of the thickness loss in non-uniformly corroded specimens is a serious technical challenge. One approach is to examine the cross-section of the specimen under a microscope, measure the area of remaining material and calculate an equivalent material thickness loss. That approach is adequate for artificially corroded material grown uniformly in the laboratory, but natural corrosion is not uniform and many such microscope studies would be required for each specimen. Such a study would be very costly and there is still a lack of universal agreement on the use of equivalent material thickness loss (Schutz, 1995). Harmsworth (1961), for example, described the effect of corrosion pitting in 2024T4 aluminum as causing stress raisers.

One problem with using equivalent material thickness loss to quantify the effect of corrosion is that there is no reliable way to measure the thickness loss during PDM. Furthermore, since corrosion occurs non-uniformly the equivalent material thickness loss would not be constant over a large skin panel. Therefore it is important to investigate alternative ways to quantify the effects of undetected corrosion on the fatigue and fracture properties of structural materials in aging aircraft. There are two variables that must be modeled as a function of corrosion damage: random crack growth for damage tolerance analysis and residual strength for fail-safe analysis. Random crack growth has already been analyzed in uncorroded and artificially corroded material (Luzar, 1997), (Whaley, 1998). Static strength properties in uncorroded and naturally corroded materials are the objective of research described in this paper.

Yield and ultimate strength were measured on corroded and uncorroded fuselage skin panels obtained from four different aircraft during PDM. The skin panels were removed from the lower fuselage below the cargo door. The identification codes used in presenting the data are described in Table 1.

Table 1. Aged Material Source Table

Code	Tail Number	Description
A	57-1502	Supplied corroded and uncorroded specimens. Entered the fleet in 1958
B	58-0010	Supplied corroded and uncorroded specimens. Unusually severe corrosion. Entered the fleet in 1959
C	60-0363	Supplied uncorroded specimens. Entered the fleet in 1962
D	57-1488	Supplied uncorroded specimens. Entered the fleet in 1958

The service histories of all four aircraft were examined. Aircraft B is the only one with unusually severe corrosive environments in its past. All the corroded materials tested were 2024T3 alloy because corrosion is most common in locations on the aircraft where this alloy is used. The test procedure, instrumentation details and results are described next.

Test Procedure

Fuselage skin panels were cut into standard ASTM E8 rectangular tension specimens eight inches long, half an inch wide with a two inch gage length. Corroded specimens were prepared by cutting between the spot welds with a band saw and then final machining with a milling machine. A load cell was used to measure load and an extensometer was used to measure deflection. The strain-rate was $1.667 \times 10^{-3} \frac{\text{in in}}{\text{sec}}$ for all specimens. The yield stress was measured by the 0.2% offset method. A power law regression analysis in the region of yielding was used to solve for the intersection with the 0.2% offset stress. The tensile strength was calculated by dividing peak load by the original specimen cross-sectional area. Modulus of elasticity was measured by a standard linear regression analysis in the elastic region.

Specimens in the longitudinal (L) direction (parallel with the rivet patterns) were cut between spot welds in long strips. Corroded specimens were cut from these larger strips so that the corrosion was in the middle of the gage length. Specimens from the long transverse (LT) direction (perpendicular to the rivet patterns) did not always guarantee the corrosion in the middle of the gage length. Each specimen was given a unique identification number that was coded to aircraft tail number, corrosion condition and whether the specimen was cut L or LT. Samples of the new material were also tested to compare with strength properties of aged material. Uncorroded specimens were typically the single thickness skins. The severity of corrosion damage was estimated by the depth and area of the corrosion within the gage length.

Four qualitative categories of corrosion damage were used: very light, light, moderate and severe. Very light corrosion was defined as no visible pitting of the surface but with some corrosion by-products, sometimes resembling moisture stains. The very light corrosion condition was used because all the skin doublers had corrosion by-products and their effect on the strength was not initially known. At the end of the project, it became obvious that the presence of corrosion by-products does not indicate degradation of strength. Light corrosion was defined as light pitting of the surface without covering a significant surface area. Moderate corrosion was defined as light pitting similar to light corrosion but more widespread in terms of

the affected area. Significant quantities of corrosion by-products were observed even for light corrosion. Severe corrosion is the easiest to describe because the surfaces are covered with black, scaly residue. The pits are so deep that the original thickness of the specimen can not be accurately determined. Specimen thickness was usually measured in an uncorroded region of the specimen, sometimes in the grip area. For all the specimens tested, there was only enough cleaning of the paint and corrosion by-products to allow machining of the specimen and measurement of the specimen thickness.

Instrumentation Details

A standard tensile test machine was used to apply load to the specimens at a constant crosshead speed. Width and thickness of every specimen was measured and recorded just before testing. Stress was measured using a load cell and elongation of the specimen was measured using an extensometer. The load cell and extensometer were based on strain gage instrumentation so they were both balanced at the beginning of each test to minimize the effect of bias error. The extensometer had a gage length of 2 inches and travel of 25% consistent with ASTM E8. A micrometer with a resolution of 0.0001 inch was used to measure the width and thickness of each specimen.

Data were collected using a PC with a 16-bit A/D acquisition card. An analysis of the uncertainty in the data from instrumentation noise was conducted by measuring a known load and deflection and comparing the variation in the data. The coefficient of variation of the stress is on the order of 10^{-4} and the coefficient of variation of the strain is on the order of 10^{-5}. Typical coefficients of variation in the yield and ultimate stress were at least two orders of magnitude greater than the instrumentation variation.

Load-deflection data was imported into a spreadsheet and the stress-strain curve was determined for each test. The instrumentation included a peak load output that was used to calculate the ultimate tensile strength. The engineering strain is the amount of elongation divided by the original gage length and is assumed to be constant over the gage length.

$$\varepsilon_e = \frac{\text{change in length}}{\text{original length}} \tag{1}$$

The engineering stress is the load divided by the original cross-sectional area.

$$\sigma_e = \frac{\text{load}}{\text{width} * \text{thickness}} \tag{2}$$

The yield stress was calculated by the intersection of the 0.2% offset stress with the engineering stress-strain curve. The ultimate stress was calculated by dividing the peak load by the original cross-sectional area, according to ASTM E8. The true strain is the integral of the infinitesimal increment (Boresi, et. al., 1993):

$$\varepsilon_t = \int \frac{dL}{L} = \ln(1 + \varepsilon_e) \tag{3}$$

The true stress is the load divided by the actual cross-sectional area under load:

$$\sigma_t = \sigma_e e^{\varepsilon_t} \tag{4}$$

The true stress-strain curve for each specimen was plotted on the same graph as the stress-strain curves from the same panel and same conditions. One of the questions to be answered by this research is whether there is a degradation of material strength due to aging without corrosion. The other question to be answered is the effect of corrosion on the yield and ultimate strength. Those results are described next.

Results and Discussion

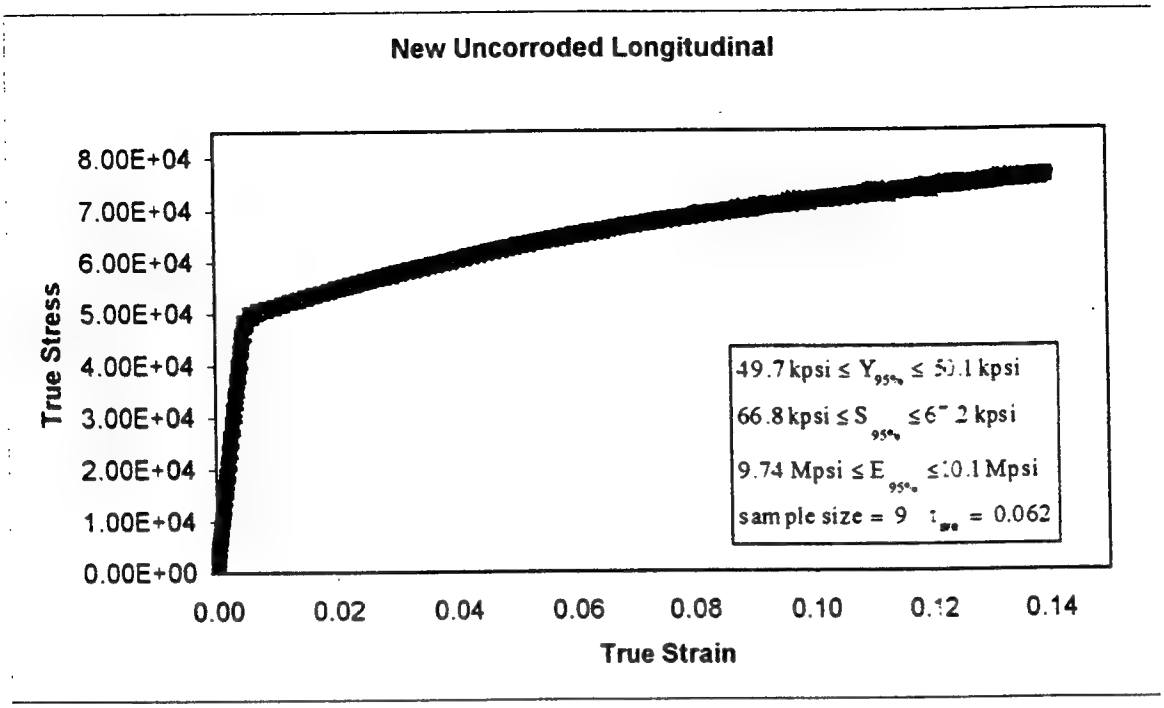
Since there is variation in material properties from one panel to the next, care was taken to assure that statistical analysis was conducted on specimens taken from the same panel. The yield and ultimate stresses were calculated for every specimen and then the 95% confidence intervals were calculated for each alloy and corrosion condition. The results of this statistical analysis are listed on every stress-strain plot along with the sample size.

Uncorroded Specimens

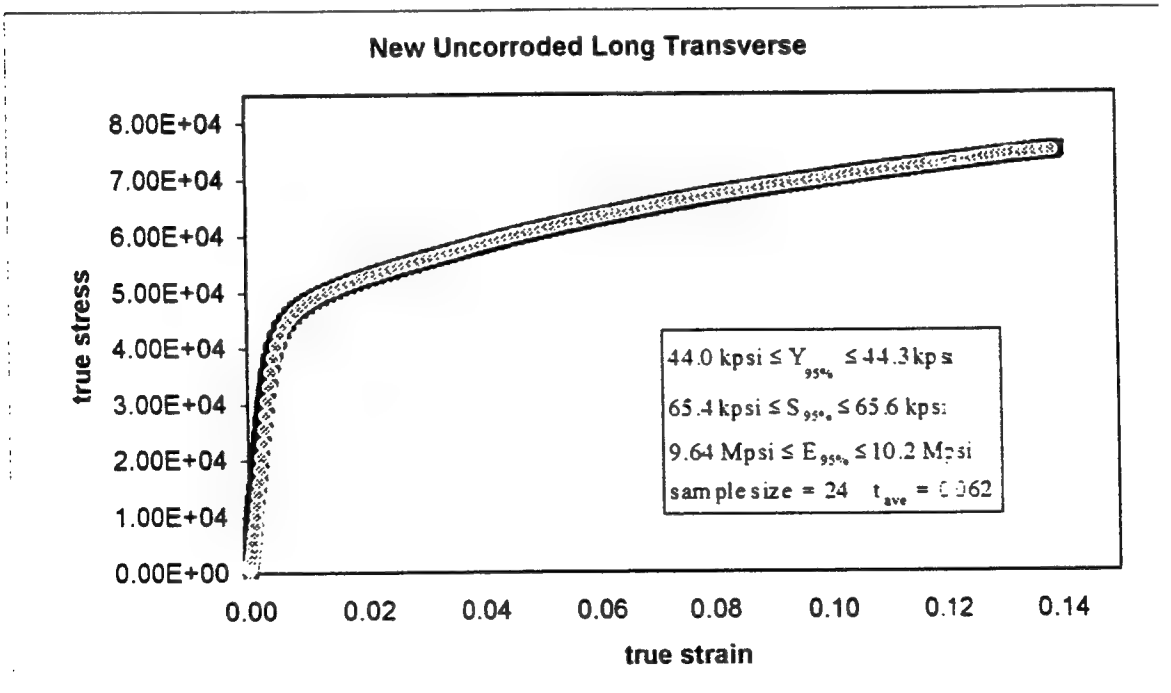
New and uncorroded aged specimens were tested first to establish a baseline. The stress-strain curves for new material are shown in Figure 1. The stress-strain curves of all the specimens are plotted together to illustrate the predictable nature of the static strength behavior. In fact, these stress-strain curves were a valuable check on the validity of the data. If any stress-strain curve varied significantly from the rest, there was usually an error in measuring the specimen thickness or in balancing the extensometer. Figure 17 in the Appendix shows an exception that is believed to be caused by localized elongation near corrosion. Figure 2 shows stress-strain curves for specimens prepared from two different panels permitting evaluation of variation between panels. The confidence intervals are all very small, 400 psi or less, suggesting very little variation in static strength properties of the new material. All of the yield and ultimate stress results compare favorably with the A-Basis values published in MIL HDBK 5. The two statistical measures used

in MIL HDBK 5 are the A-Basis and the B-Basis. At least 99 percent of the population of values is expected to equal or exceed the A-Basis property with 95 percent confidence. At least 90 percent of the population of values is expected to equal or exceed the B-Basis property with 95 percent confidence. The A-Basis properties from MIL HDBK were chosen for comparison to these results because they are more conservative. For 2024T3 sheet, L direction, the A-Basis properties are: $F_{tu} = 64$ kpsi and $F_{ty} = 47$ kpsi. For the LT direction, the A-Basis properties are: $F_{tu} = 63$ kpsi and $F_{ty} = 42$ kpsi. All the new specimens exceeded the A-Basis property values, as expected.

Figures 3-6 show the uncorroded stress-strain curves of the four populations summarized in Table 1. There is significantly more variation in the strength of the aged material. Some panels had confidence intervals around 200 psi while others had confidence intervals as high as 1300 psi. The mean stress also varied significantly from panel to panel. Reduced variation in properties of the new material is probably due to improved quality control in manufacturing and not due to any effect of aging in the old material (Bucci and Warren, 1997). For most uncorroded specimens tested, the yield and ultimate strengths of the aged material exceeded the A-Basis properties from MIL HDBK 5. Two hundred seventy eight uncorroded specimens from four different aircraft were tested and compared with 82 new specimens. There was no significant difference in strength of the aged specimens compared with the new ones. These results show that the strength of the aged, uncorroded material is not degraded over time.

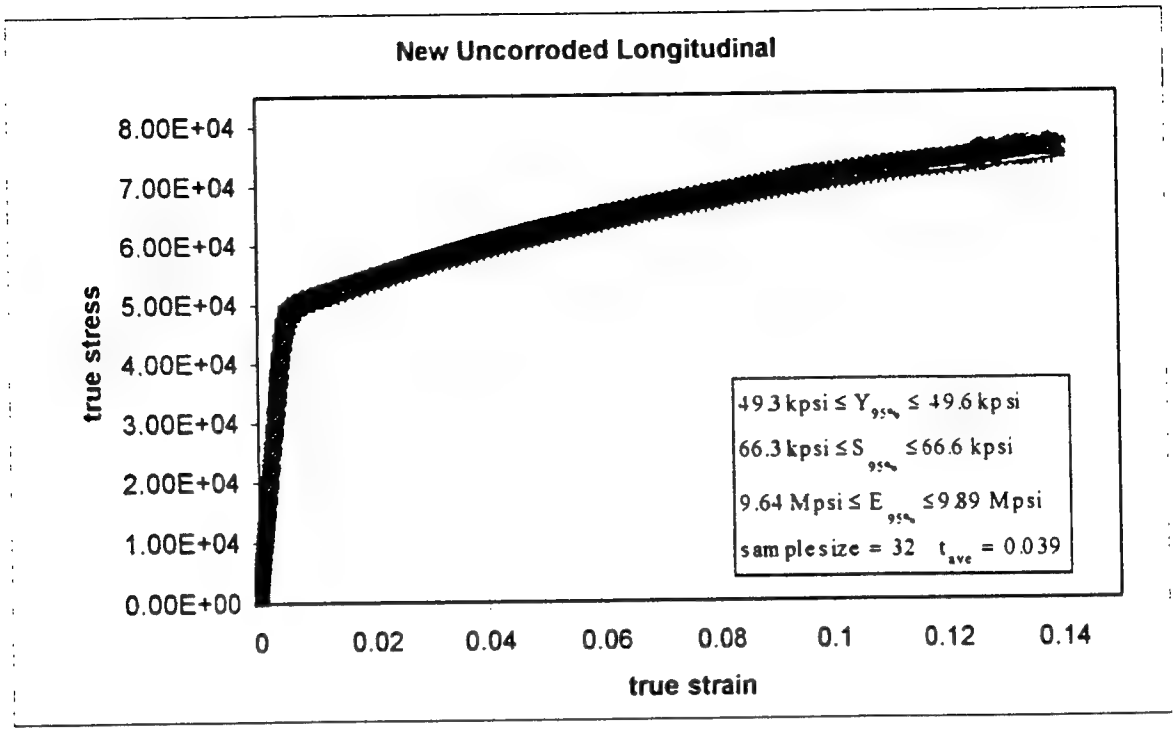


a. Longitudinal: From MIL HDBK 5, Ftu = 64 kpsi; Fty = 47 kpsi

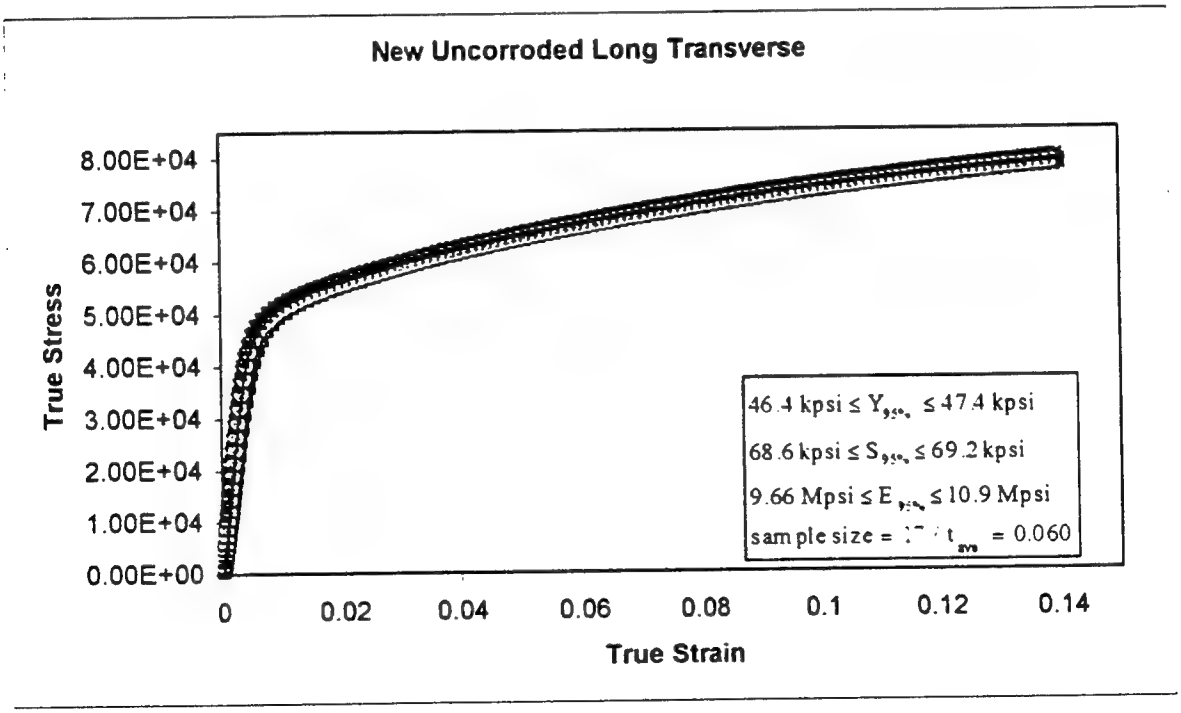


b. Long Transverse: From MIL HDBK 5, Ftu = 63 kpsi; F_{ty} = 42 kpsi

Figure 1. Stress-Strain Curves for New Material, L and LT Directions

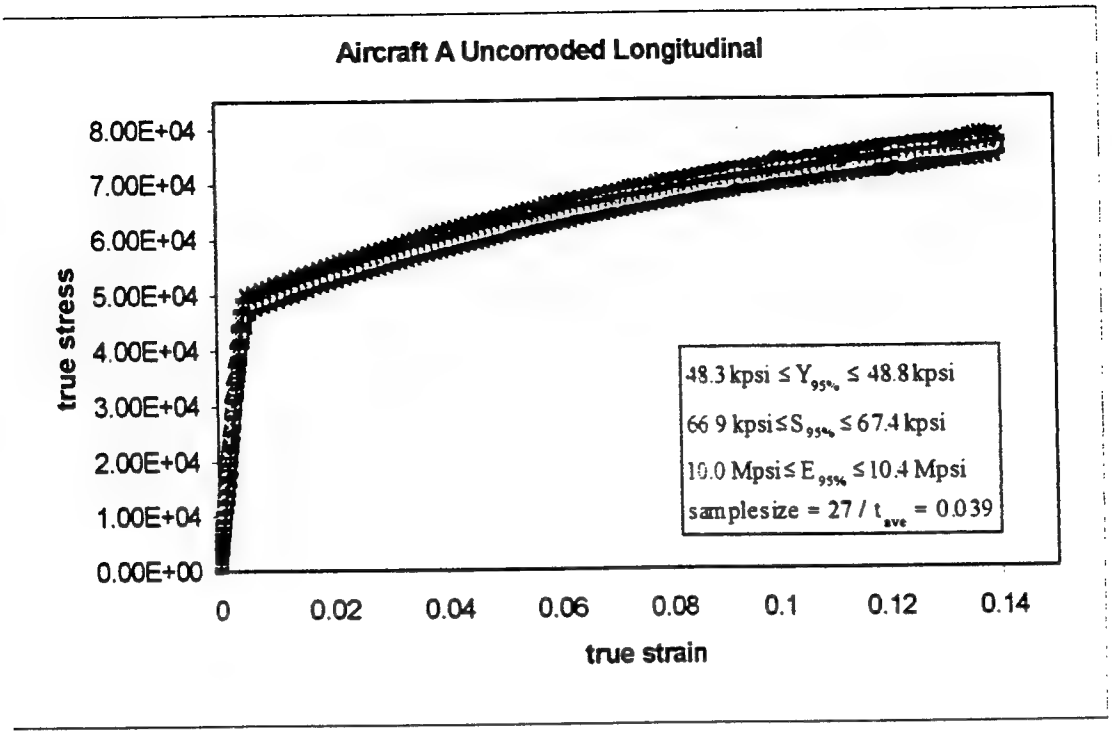


a. Longitudinal: From MIL HDBK 5, Ftu = 64 kpsi; Fty = 47 kpsi

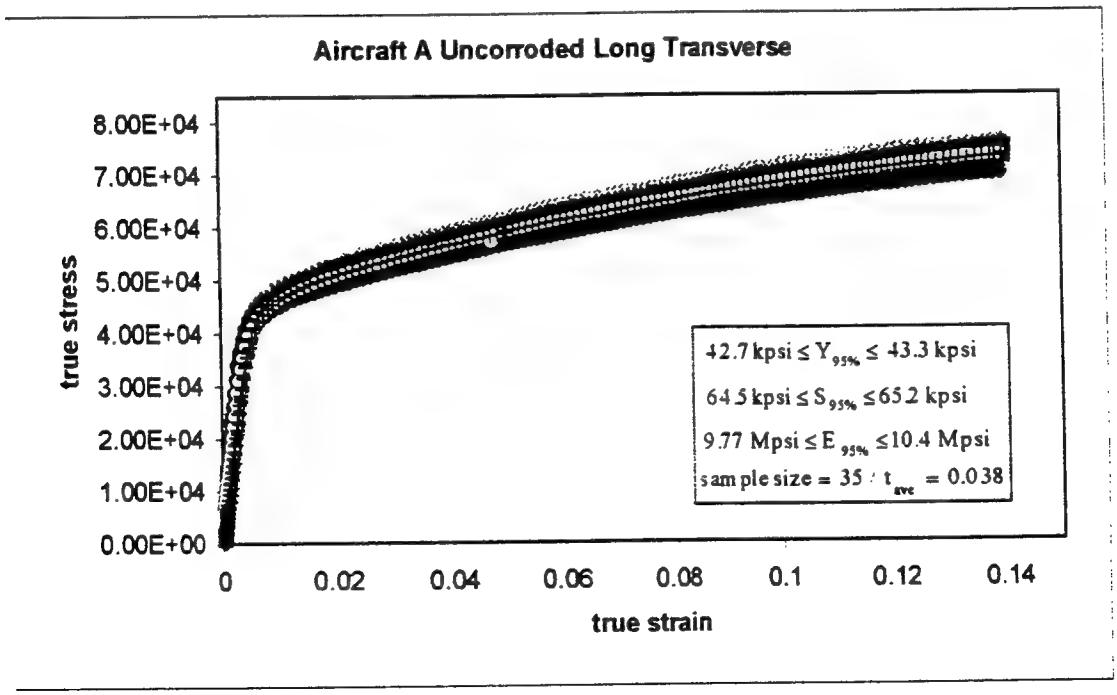


b. Long Transverse: From MIL HDBK 5, Ftu = 63 kpsi; F_{ty} = 42 kpsi

Figure 2. Stress-Strain Curves for New Material, L and LT Directions

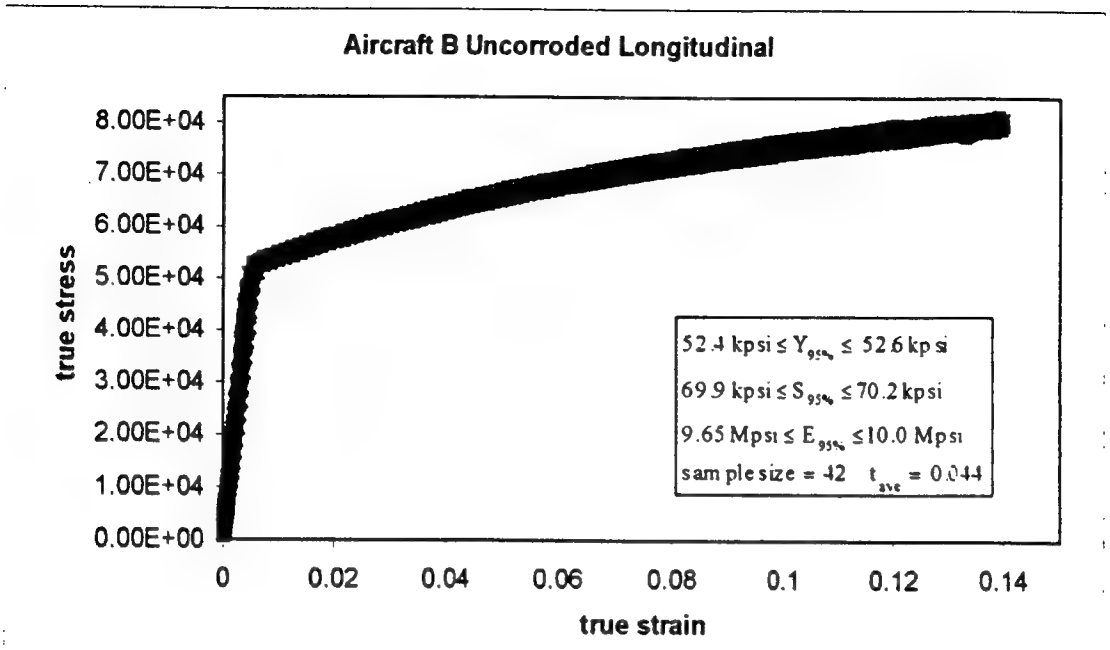


a. Longitudinal: From MIL HDBK 5, Ftu = 64 kpsi; Fty = 47 kpsi

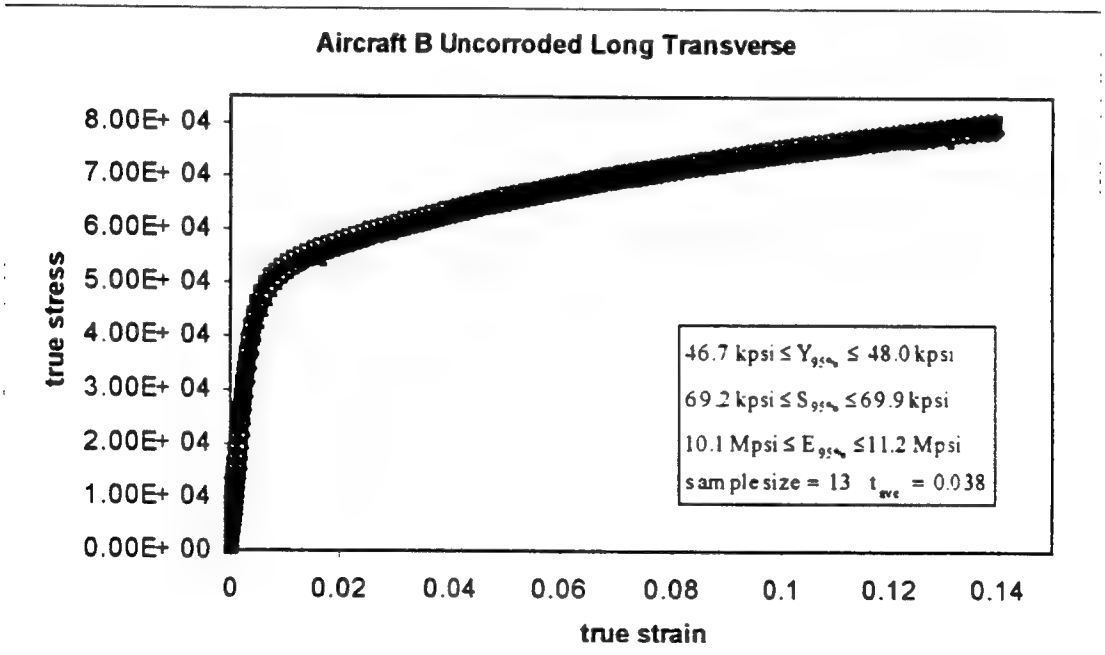


b. Long Transverse: From MIL HDBK 5, Ftu = 63 kpsi; F_{ty} = 42 kpsi

Figure 3. Stress-Strain Curve for 2024T3, Aged Uncorroded Material, L and LT Directions, Aircraft A

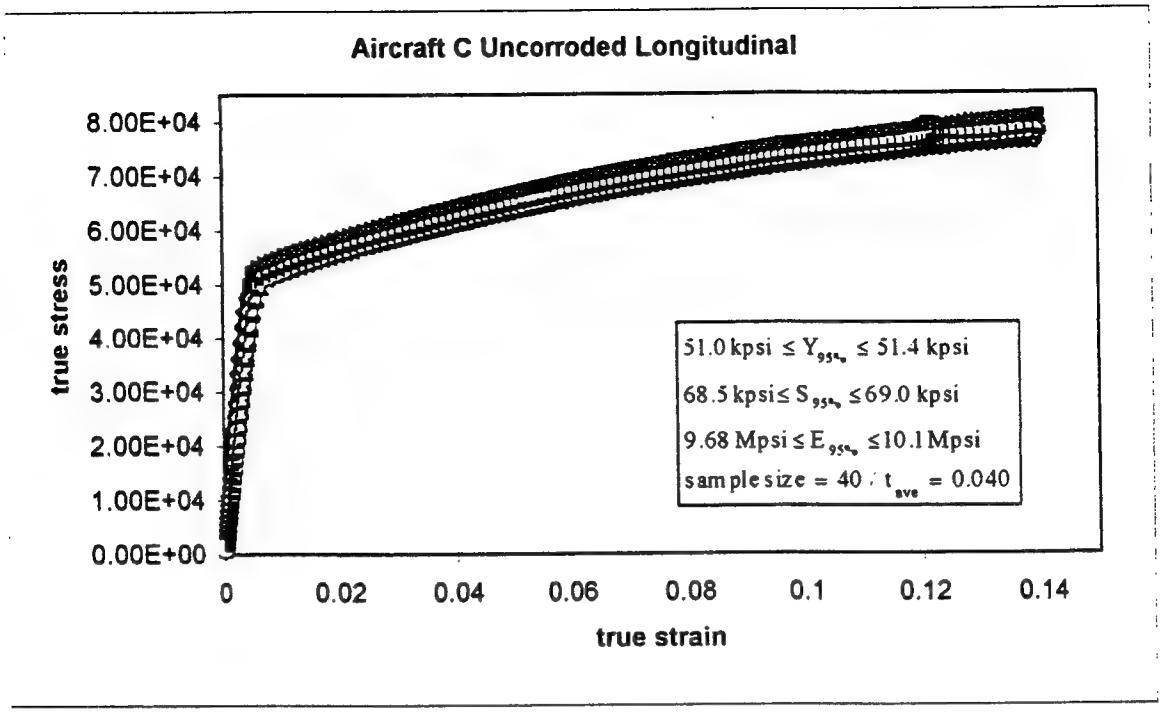


a. Longitudinal: From MIL HDBK 5, Ftu = 64 kpsi; Fty = 47 kpsi

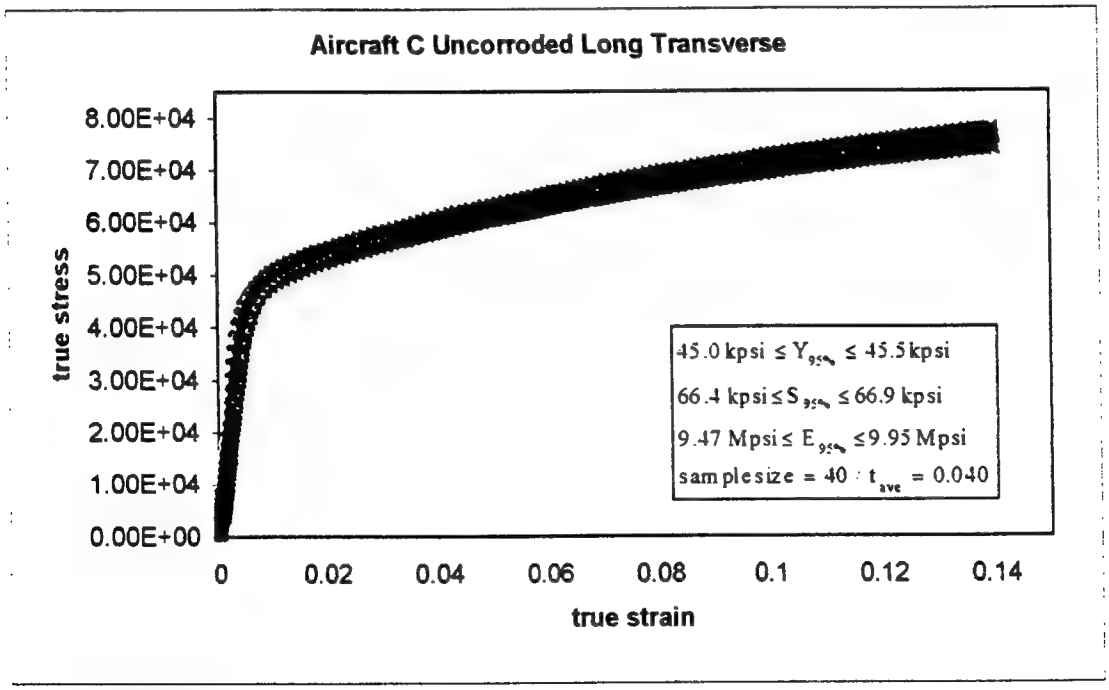


b. Long Transverse: From MIL HDBK 5, Ftu = 63 kpsi; F_{ty} = 42 kpsi

Figure 4. Stress-Strain Curve for 2024T3, Aged Uncorroded Material, L and LT Directions, Aircraft B

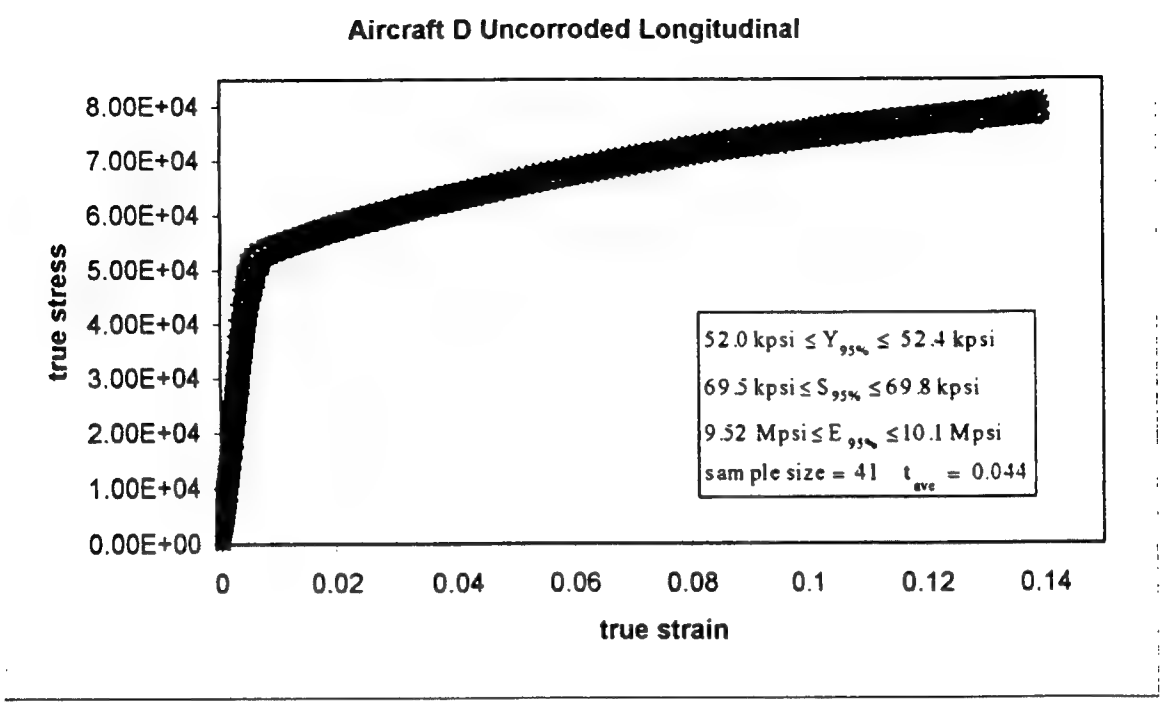


a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi, F_{ty} = 47 kpsi

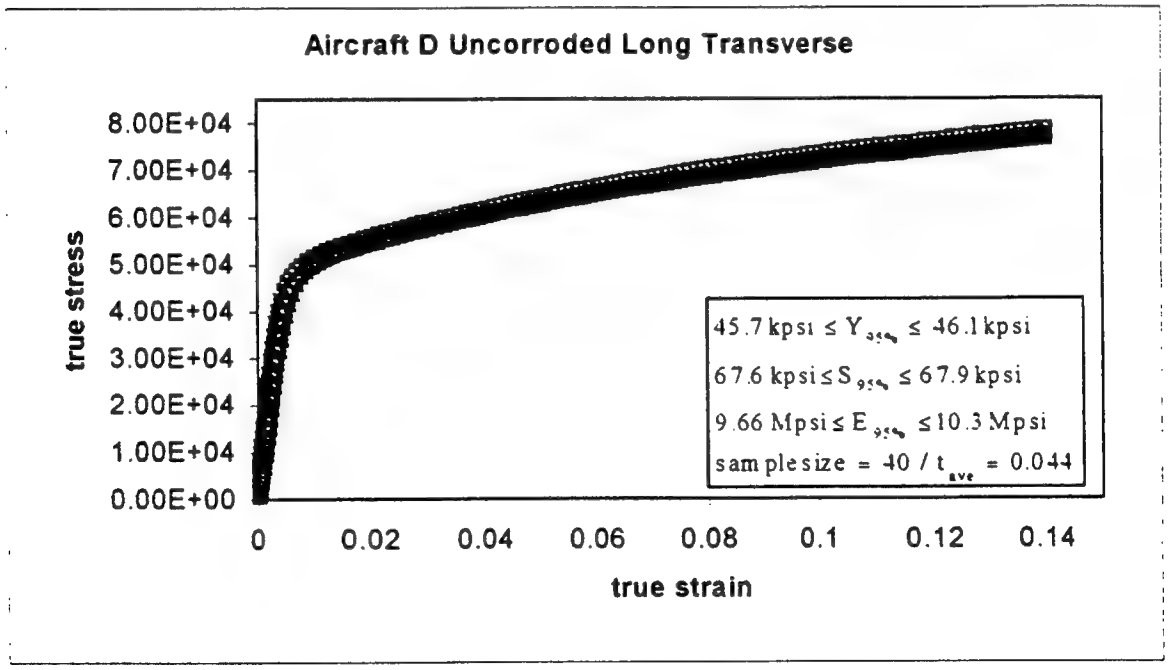


b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi, F_{ty} = 42 kpsi

Figure 5. Stress-Strain Curve for 2024T3, Aged Uncorroded Material, L and LT Directions, Aircraft C



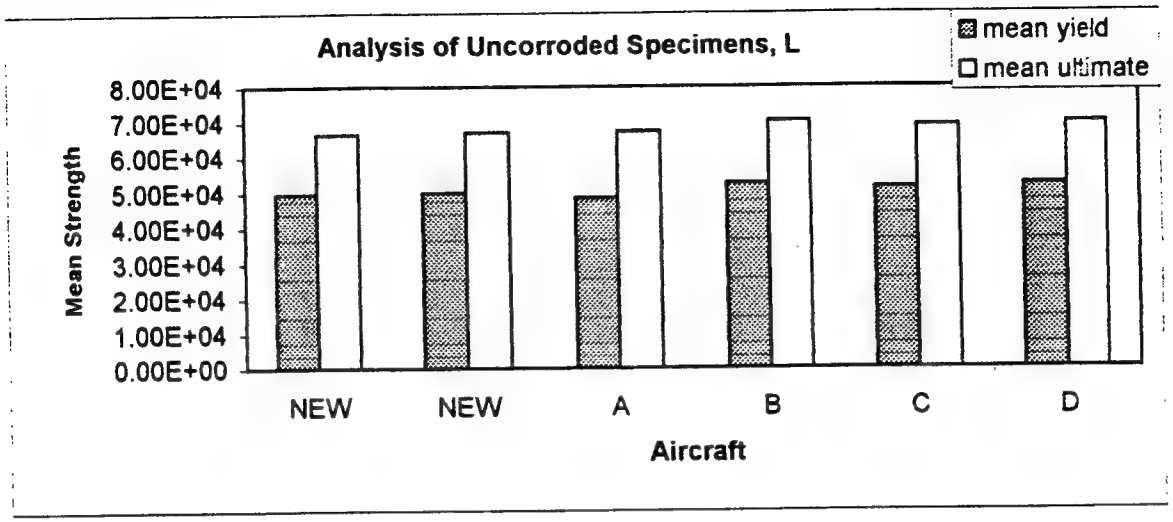
a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi, F_{ty} = 47 kpsi



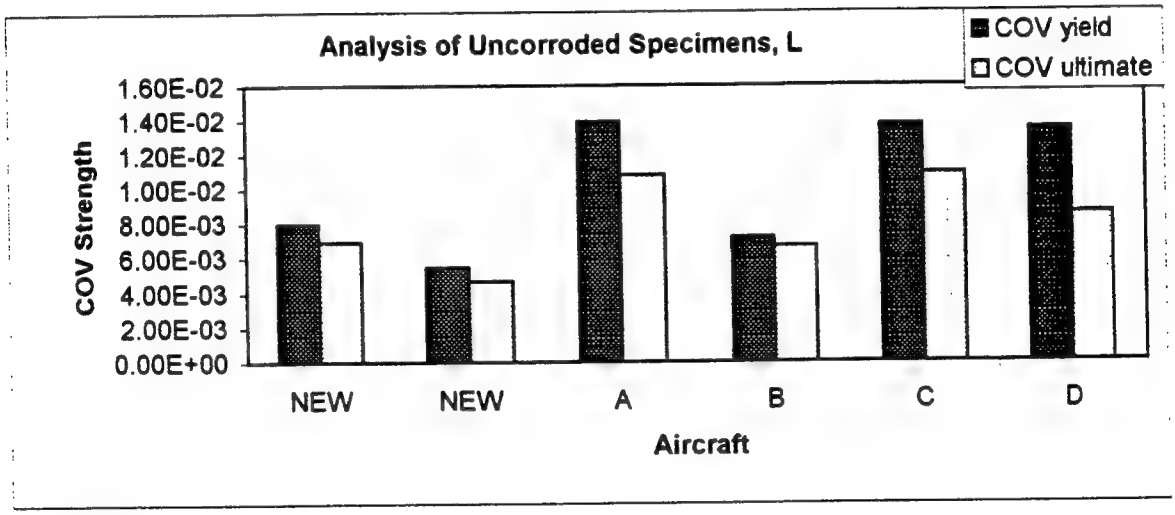
b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi, F_{ty} = 42 kpsi

Figure 6. Stress-Strain Curve for 2024T3, Aged Uncorroded Material, L and LT Directions, Aircraft D

Figure 7 shows the means and coefficients of variation of yield and ultimate strength of the two populations of new material and the aged material taken from the four aircraft. Figure 7a shows that the mean strength of the aged material is as good as the new. Figure 7b illustrates the reduced variation in strength for the new material.



a. Mean Strength



b. Coefficients of Variation

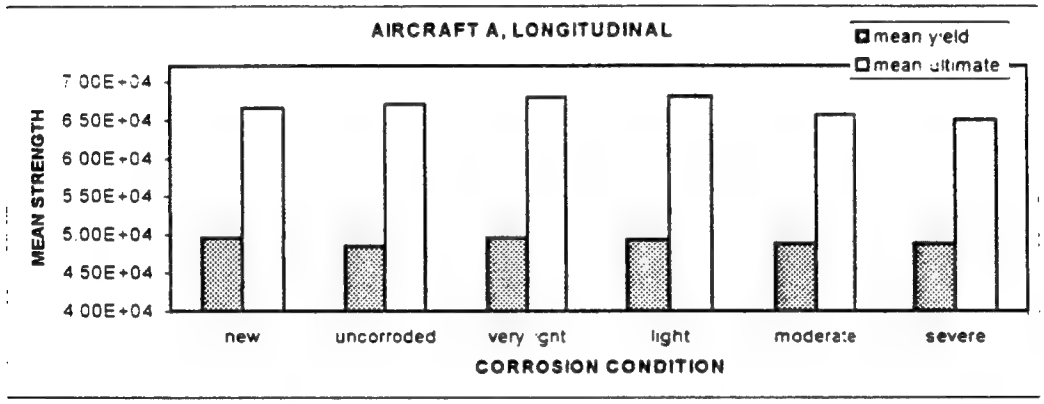
Figure 7. Mean and Coefficients of Variation of Uncorroded New and Aged 2024T3 Aluminum

Corroded Specimens

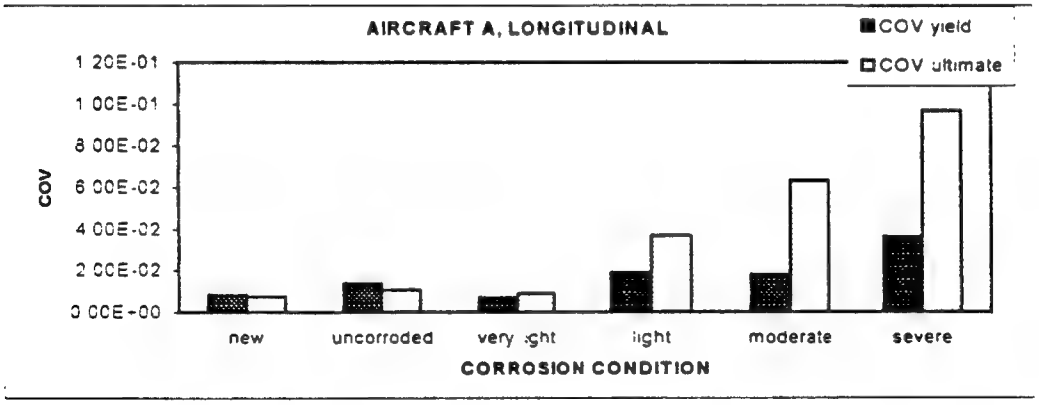
Corrosion was always found somewhere between skin doublers. Even when there was no visible pitting, the white powder corrosion by-product could be seen as the specimens were being

roughed out with the bandsaw. In many cases, a large panel removed from the aircraft because of corrosion in a localized area showed significant corrosion throughout the panel. There was no distinct correlation between elastic modulus and corrosion condition. Two hundred seventy one corroded specimens were tested and the stress-strain plots of all of them are given in the Appendix. It became apparent that the strength variation from panel to panel is greater than the variation between some corrosion conditions. Therefore comparisons of corrosion conditions must be made on specimens taken from the same panel. The specimen thickness is shown on each of the stress-strain plots so that comparisons of corrosion severity could be identified from the same population taken from the same panel.

Figure 8 shows the mean and coefficient of variation of the yield and ultimate strengths of Aircraft A as a function of corrosion severity.



a. Mean Strength

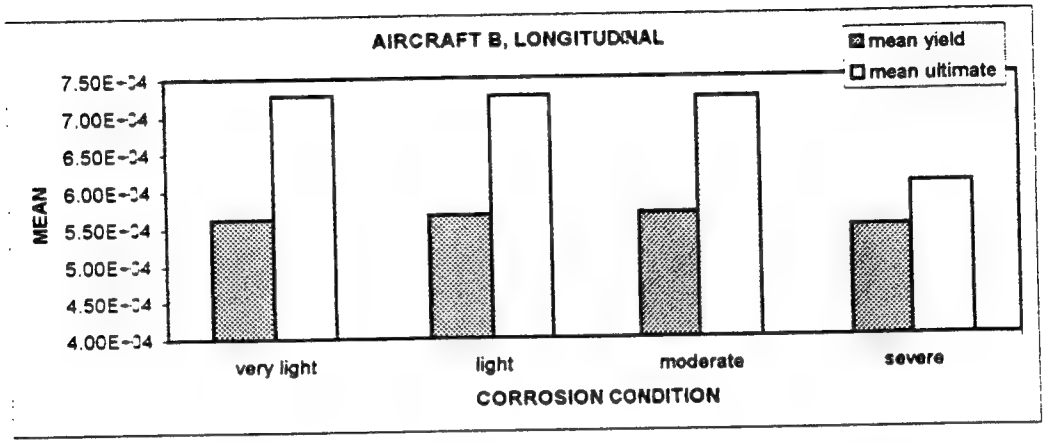


b. Strength Coefficients of Variation

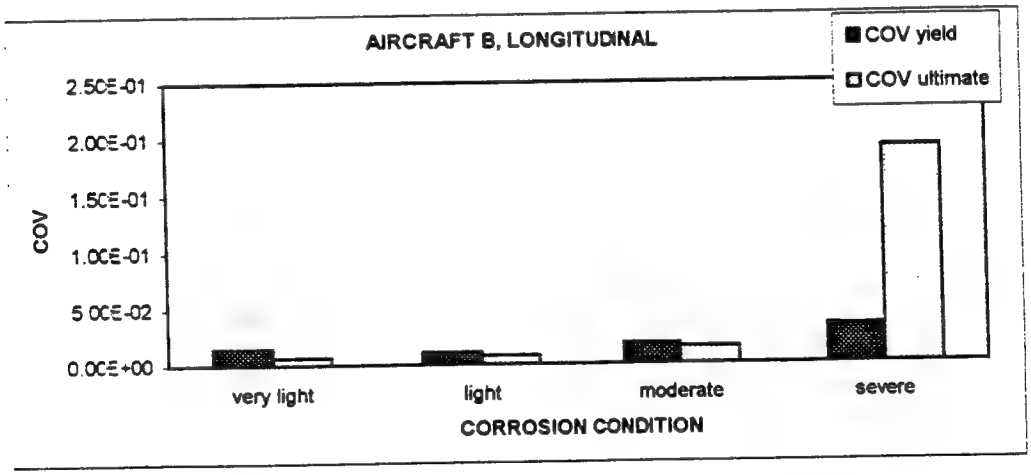
Figure 8. Analysis of Corrosion in 2024T3 Aluminum, Aircraft A

Figure 8a suggests that corrosion severity has no effect on the mean yield stress and a small effect on the mean ultimate stress. Figure 8b shows that the coefficient of variation of the strength increases as corrosion severity increases. This is logical since there was no quantitative characterization of corrosion damage in this research. Uncertainty of the thickness loss would certainly increase the variation in the strength.

Figure 9 shows the mean and coefficient of variation as a function of corrosion for aircraft B.



a. Mean Strength



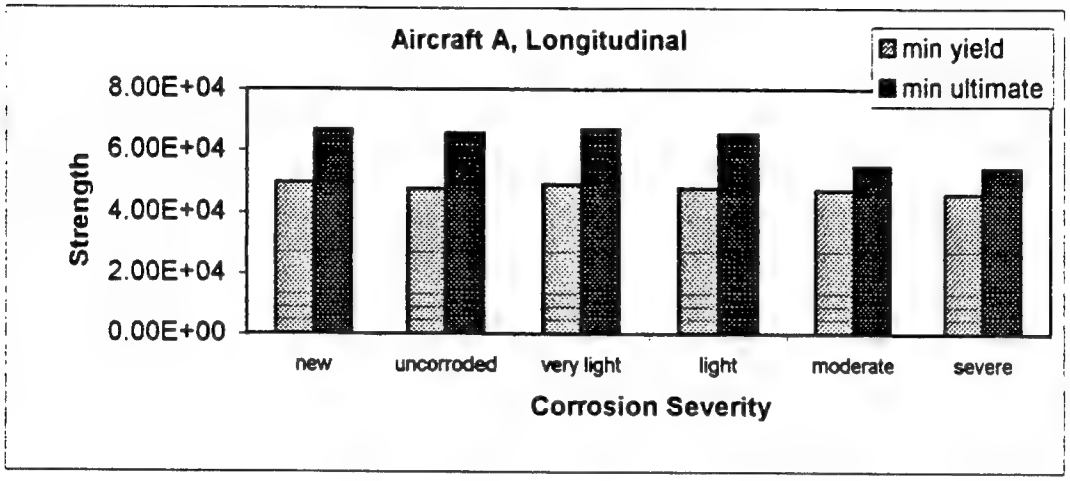
b. Strength Coefficients of Variation

Figure 9. Analysis of Corrosion in 2024T3 Aluminum, Aircraft B

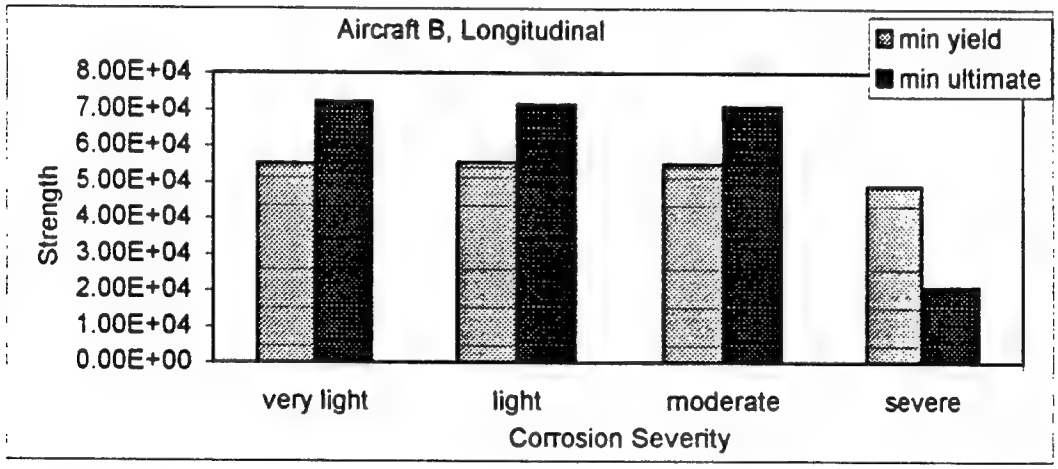
The corrosion was much more severe in aircraft B and there were many more specimens classified as severe. Since there were no uncorroded specimens from the same panel as the corroded conditions, the very light corrosion condition is used as a baseline in Figure 9. Figure 9a shows that the mean ultimate stress decreases as corrosion severity increases consistent with

the trend suggested in Figure 8a. Specimens labeled severe from aircraft B had much lower mean ultimate stress than severe specimens from aircraft A. This is another example of the need for a quantitative description of corrosion severity in naturally corroded specimens. Figure 9b shows a significant increase in variation for more severe corrosion, consistent with the trend suggested in Figure 8b.

Figures 8 and 9 show a decrease in the mean ultimate stress as a result of corrosion. The effect is more evident when the minimum stress is plotted, as in Figure 10 for both aircraft.



a. Aircraft A



b. Aircraft B

Figure 10. Minimum Yield and Ultimate Strengths of Aircraft A and B as a Function of Corrosion Severity

In all cases corrosion has a significant effect on the mean ultimate stress and a slight effect on the mean yield stress.

For moderate and severe corrosion conditions the maximum strain decreased substantially in many cases. All of the less severely corroded specimens held significant load past 15% strain. In many severe corrosion cases the maximum strain was two percent or less. The moderate corrosion condition in Figure 22b shows a specimen with a significant decrease in the maximum strain. Figure 11 illustrates this for the most severe cases tested.

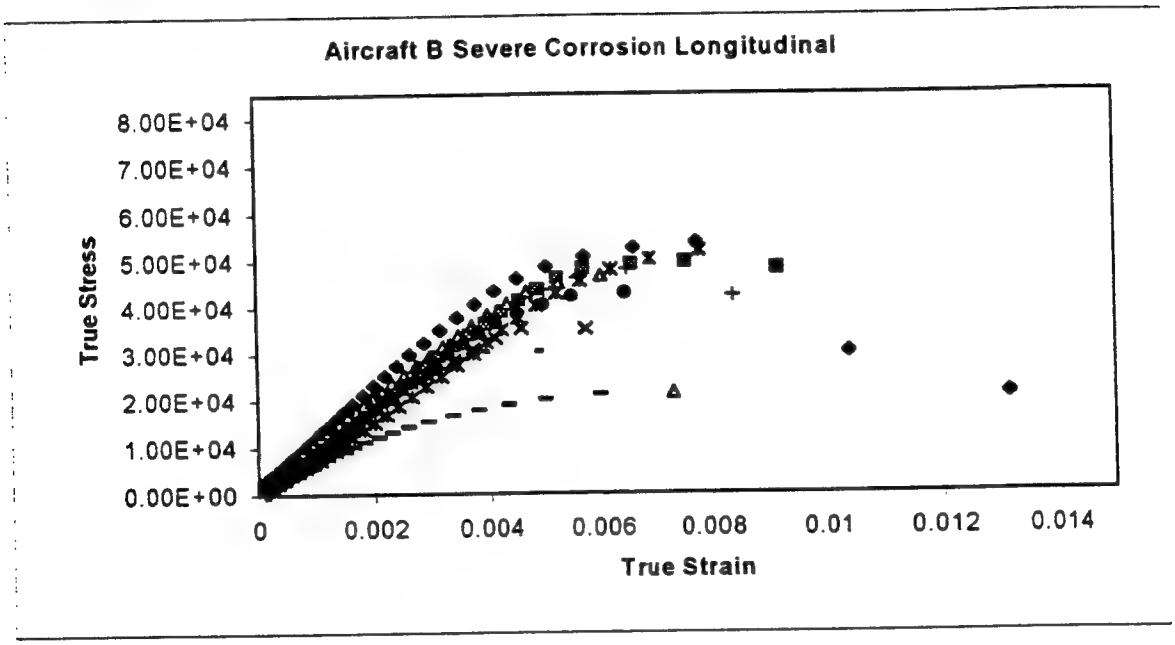


Figure 11. Analysis of Severe Corrosion in 2024T3 Aluminum, Aircraft B

All specimens in Figure 11 failed before there was enough data to calculate a yield stress. The presence of corrosion significantly decreases the plastic strain the material can support. In most cases the corroded specimens broke near corrosion, although not always in the most severe corrosion. In the severely corroded specimens, remnants of the spot welds could sometimes be seen in the gage length. Those specimens always fractured at one of the spot welds. The heat treatment is assumed to be compromised by the spot welding fabrication technique. Replacement panels are not spot welded during PDM.

Summary and Conclusions

Many researchers are measuring the effect of corrosion artificially induced in the laboratory, but this is the only research of its kind devoted to quantifying the effects of naturally occurring

corrosion in an active fleet. The most serious technical challenge to this research was the lack of a quantitative measure of corrosion damage. Most researchers use the equivalent material thickness loss for laboratory corrosion. This is appropriate since artificial corrosion grown in the laboratory can be caused to occur uniformly. Natural corrosion is not uniform. Equivalent material thickness loss would require extensive metalographic studies of several cross-sections within the gage length. Such an extensive procedure is not practical and is beyond the scope and budget of the current research. There is a great need to develop a quantitative description of corrosion damage that relates non-uniform thickness loss to degradation of structural strength properties.

Acknowledgements

Support by Don Nieser, OC-ALC, in obtaining aged aircraft materials for specimens is gratefully acknowledged.

References

Aluminum Standards and Data 1984. Eighth Edition, The Aluminum Association, Inc. Washington D. C: 1984.

Boresi, A. P., R. J. Schmidt and O. M. Sidebottom, *Advanced Mechanics of Materials*, Fifth Edition, John Wiley & Sons, New York, 1993.

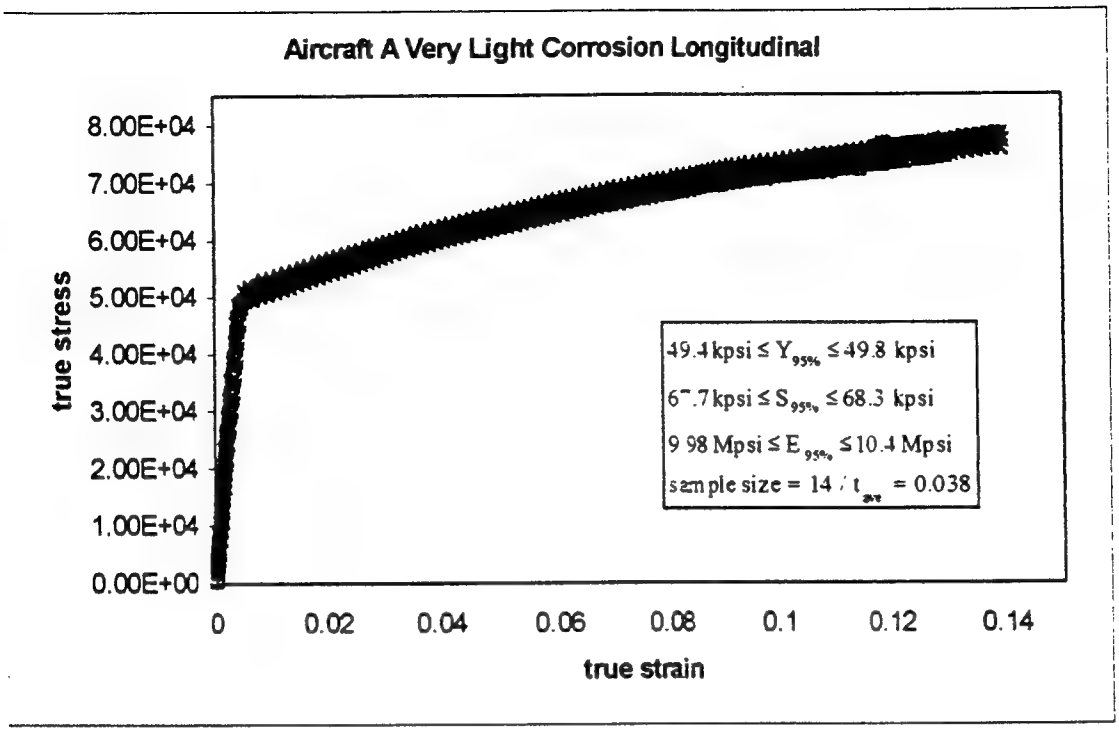
Boyer, H. E. and Gall T. L., (eds.) *Metals Handbook, Desk Edition*. American Society for Metals, Metals Park, Ohio: 1985.

Bucci, R. J., and C. J. Warren, "Material Substitutions for Aging Aircraft," Presented at the First Joint DOD/FAA/NASA Conference on Aging Aircraft, 8-10 July, 1997, Ogden Utah.

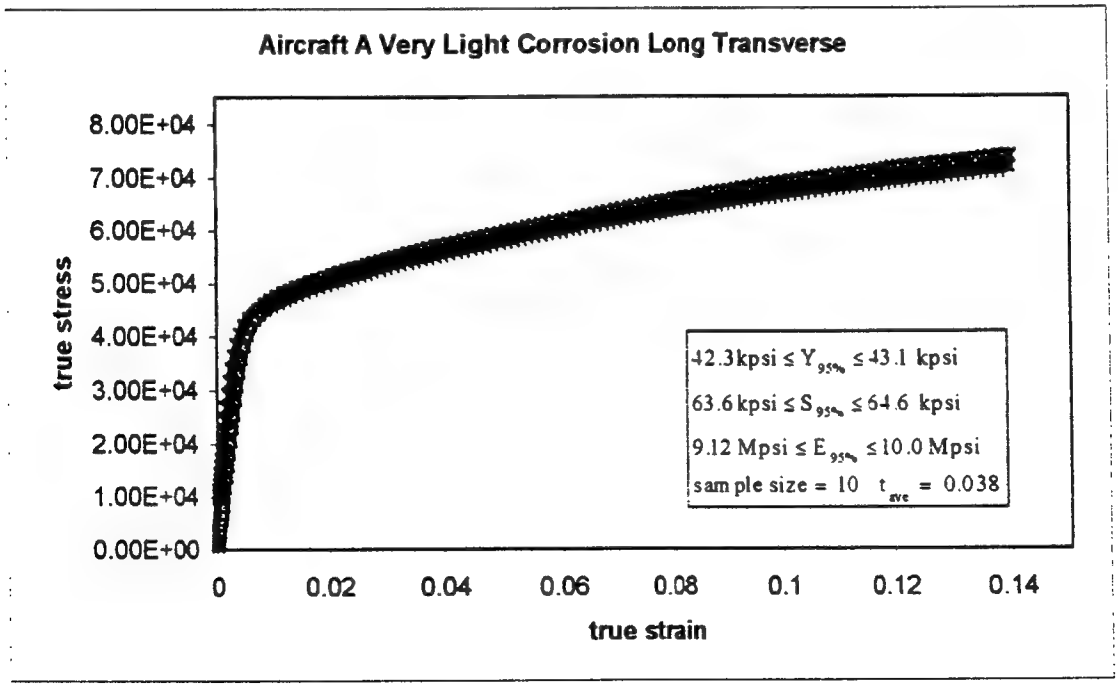
Luzar, J., "Corroded Material Crack Growth Rate Test Results," Presented at the First Joint DOD/FAA/NASA Conference on Aging Aircraft, 8-10 July, 1997, Ogden Utah.

Schutz, W., "Corrosion Fatigue – The Forgotten Factor in Assessing Durability," 15th Plantema Memorial Lecture, 18th Symposium of the international Committee on Aeronautical Fatigue, Melbourne, May 1995.

Whaley, P. W., "A Probabilistic Framework for the Analysis of Corrosion Damage in Aging Aircraft," Presented at the 39th Structures, Structural Dynamics and Materials Conference, April 20-23, 1998, Long Beach CA.

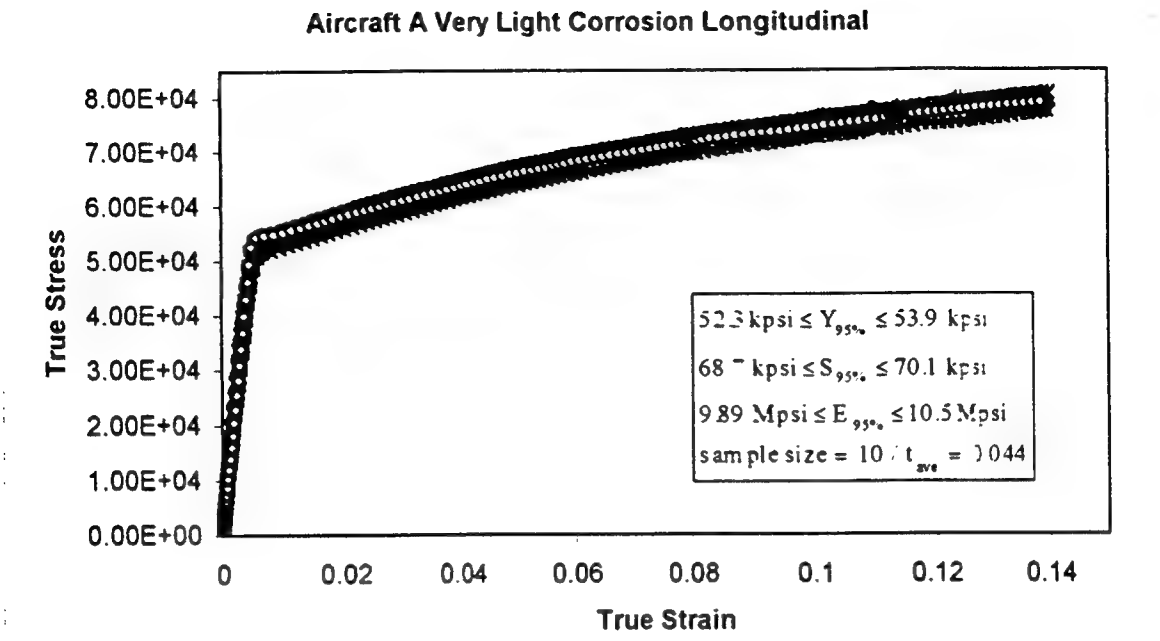


a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi; F_{ty} = 47 kpsi

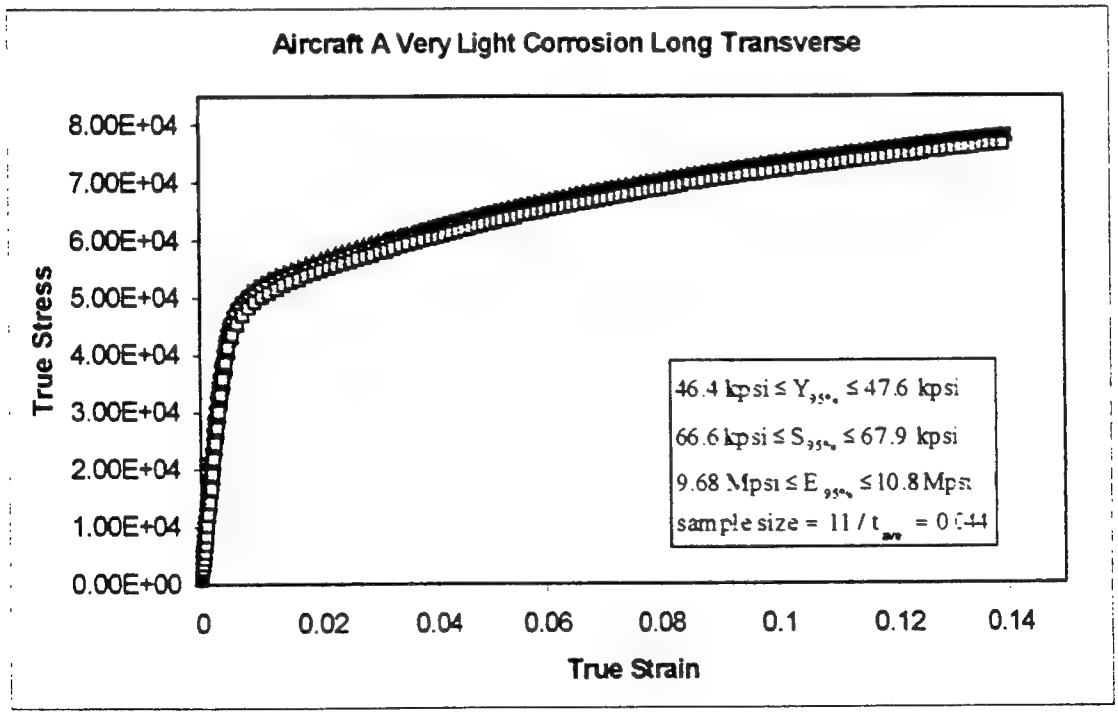


b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi; F_{ty} = 42 kpsi

Figure 12. Very Light Corrosion Condition in Aircraft A, L and LT Directions

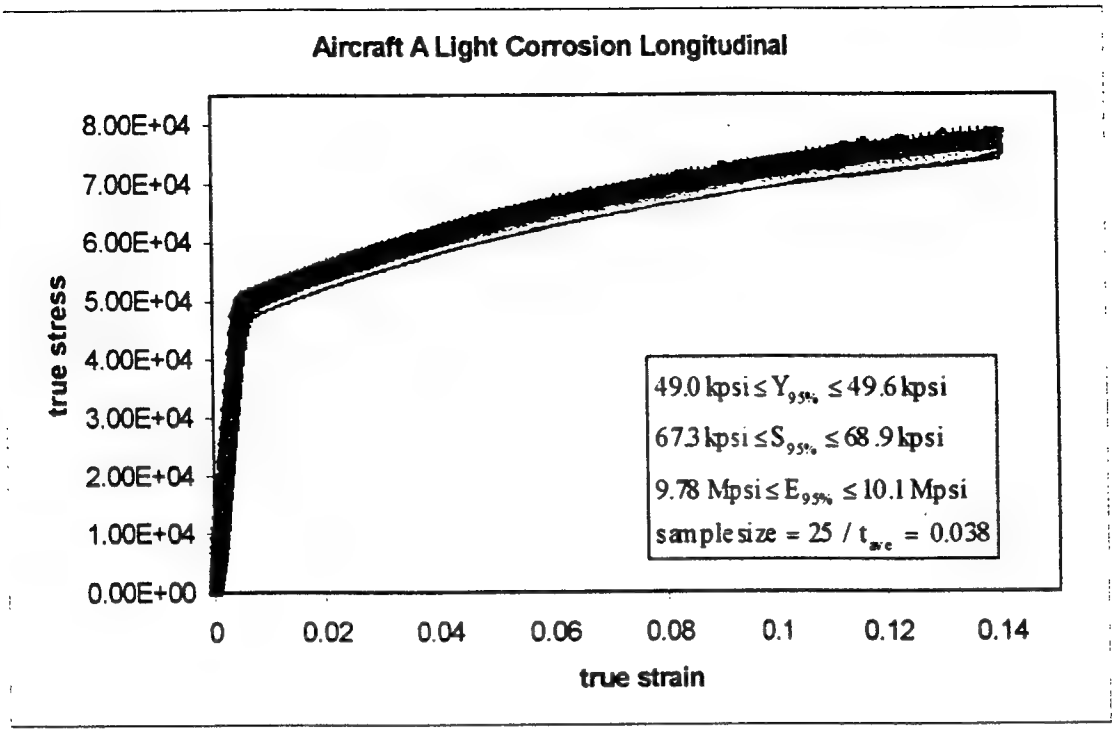


a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi; F_{ty} = 47 kpsi

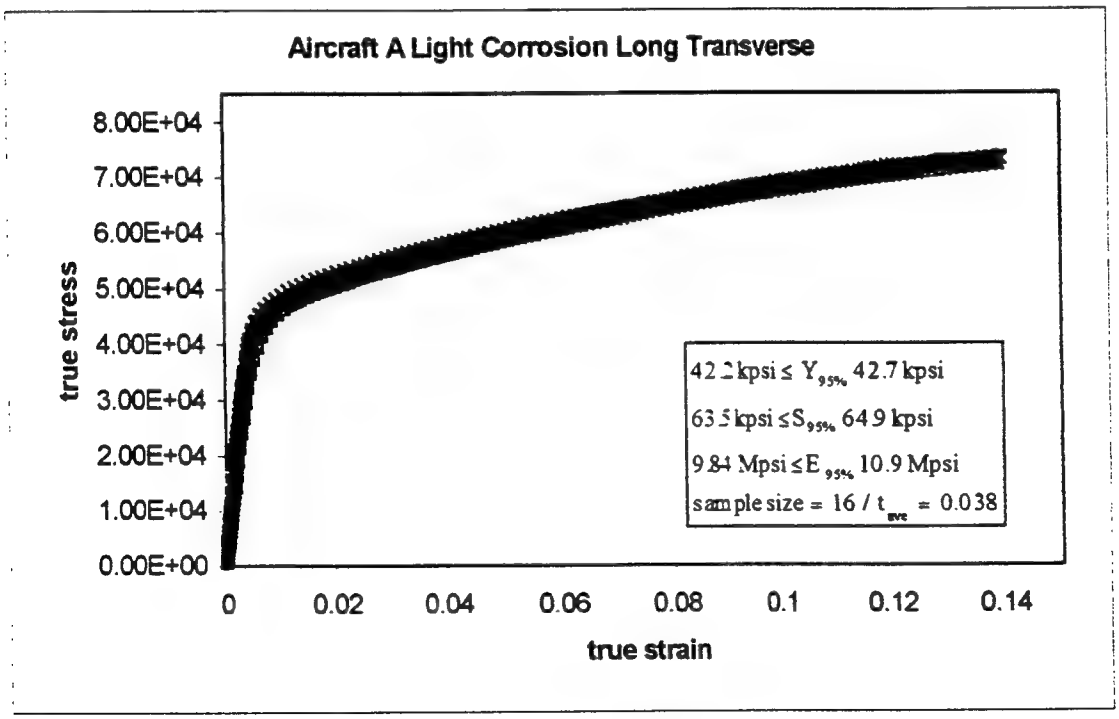


b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi; F_{ty} = 42 kpsi

Figure 13. Very Light Corrosion Condition in Aircraft A, L and LT Directions, Alternate Population

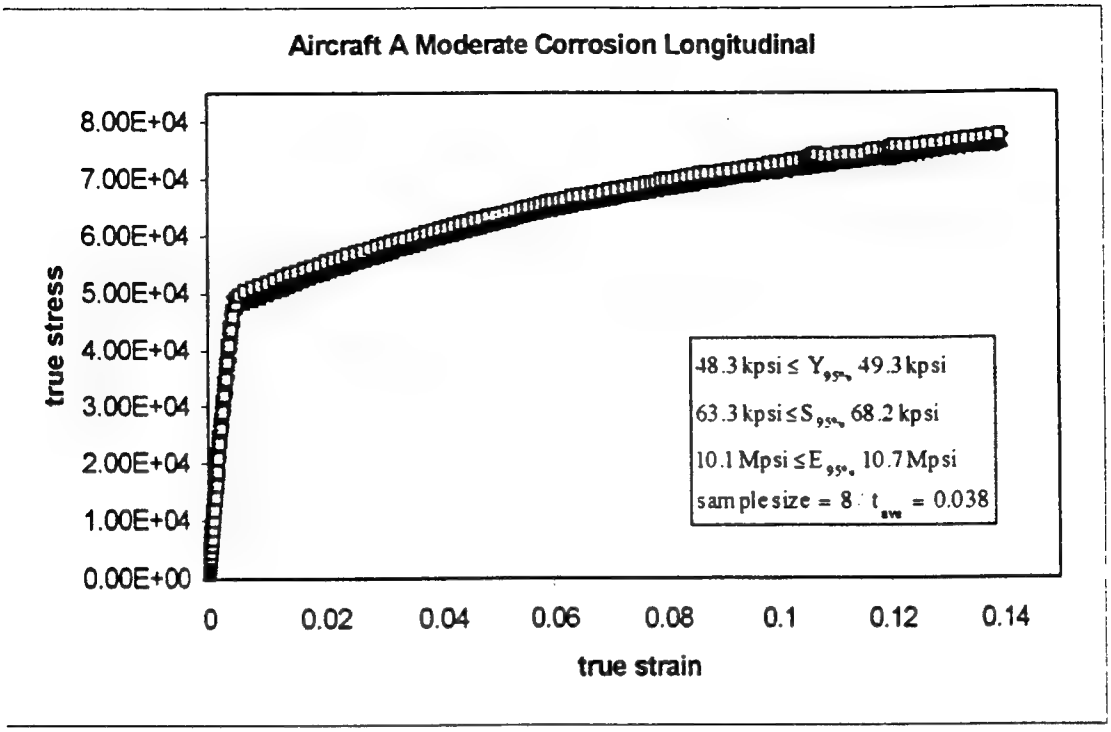


a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi; F_{ty} = 47 kpsi

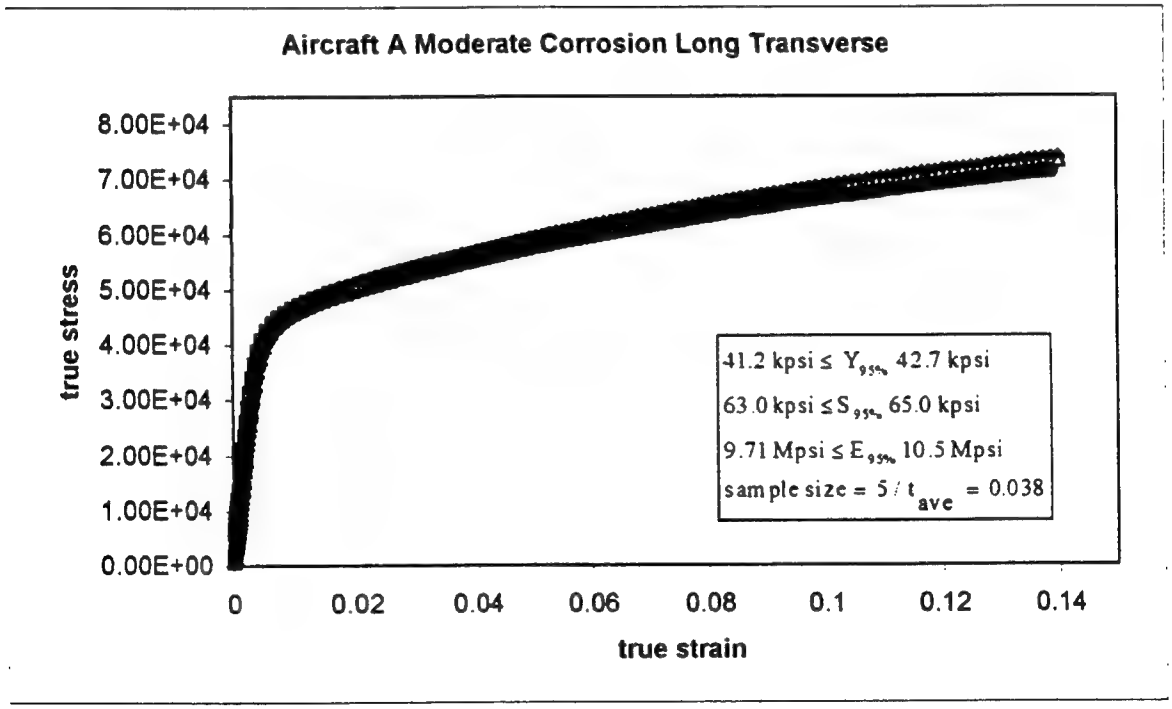


b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi; F_{ty} = 42 kpsi

Figure 14. Light Corrosion Condition in Aircraft A, L and LT Directions



a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi; F_{ty} = 47 kpsi



b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi; F_{ty} = 42 kpsi

Figure 15. Moderate Corrosion Condition in Aircraft A, L and LT Directions

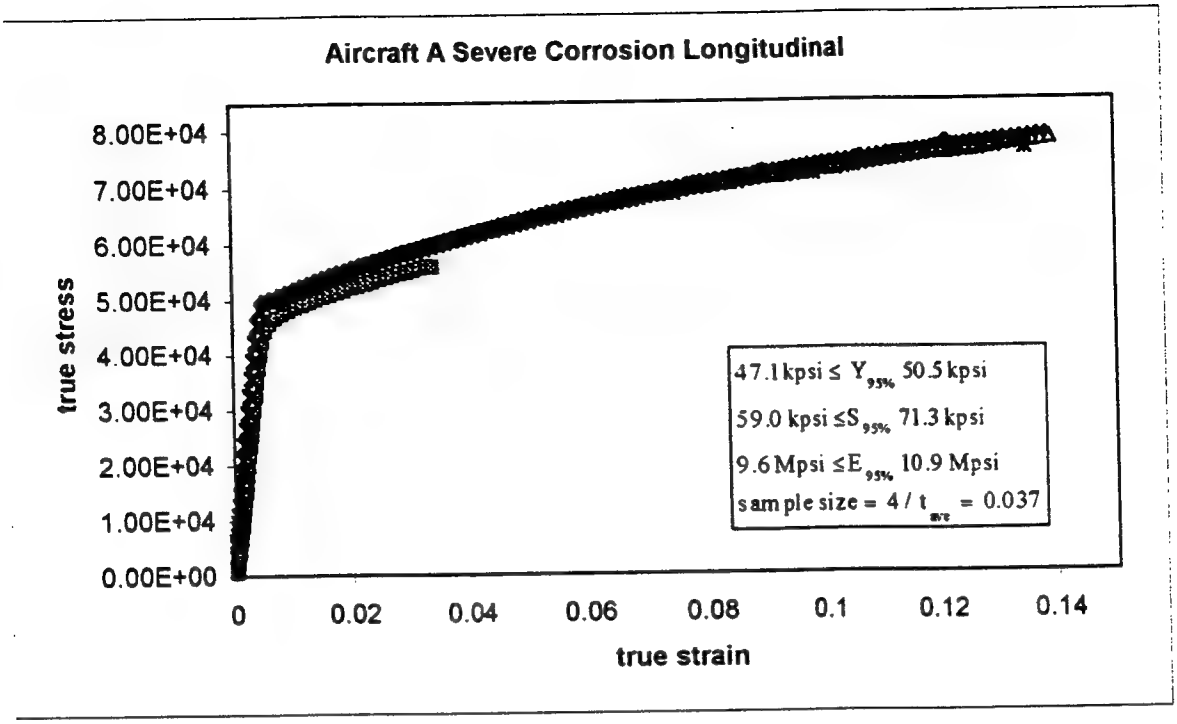


Figure 16. Severe Corrosion Condition in Aircraft A, L Direction

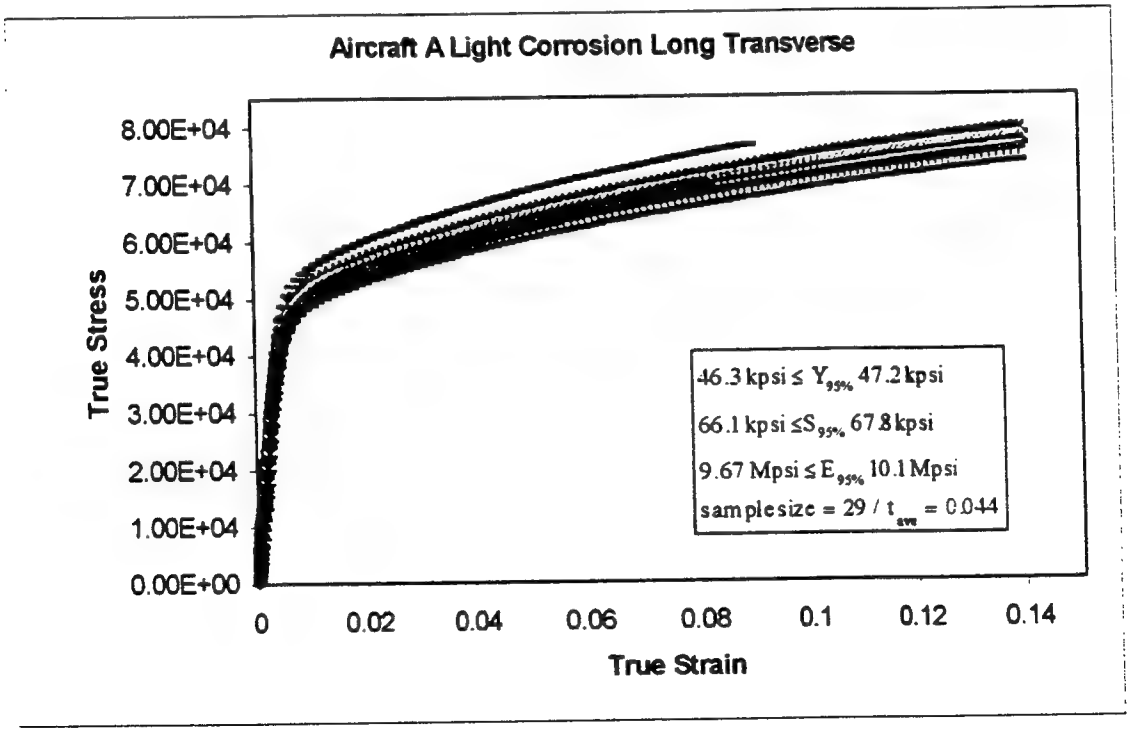
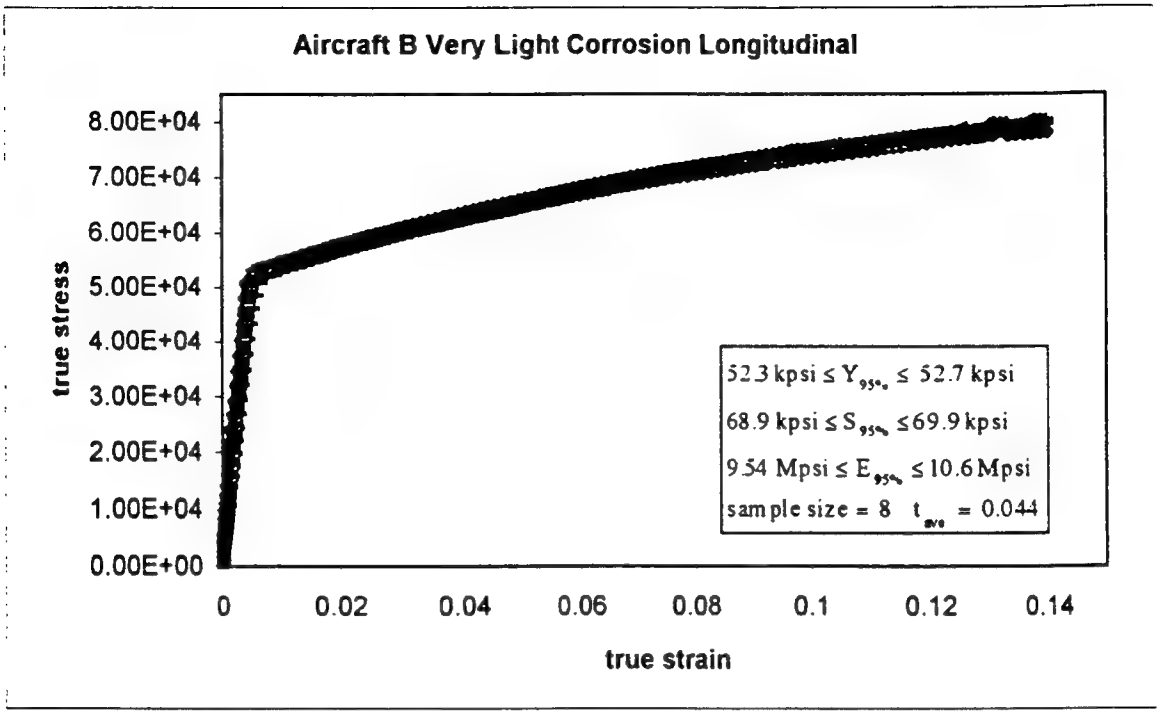
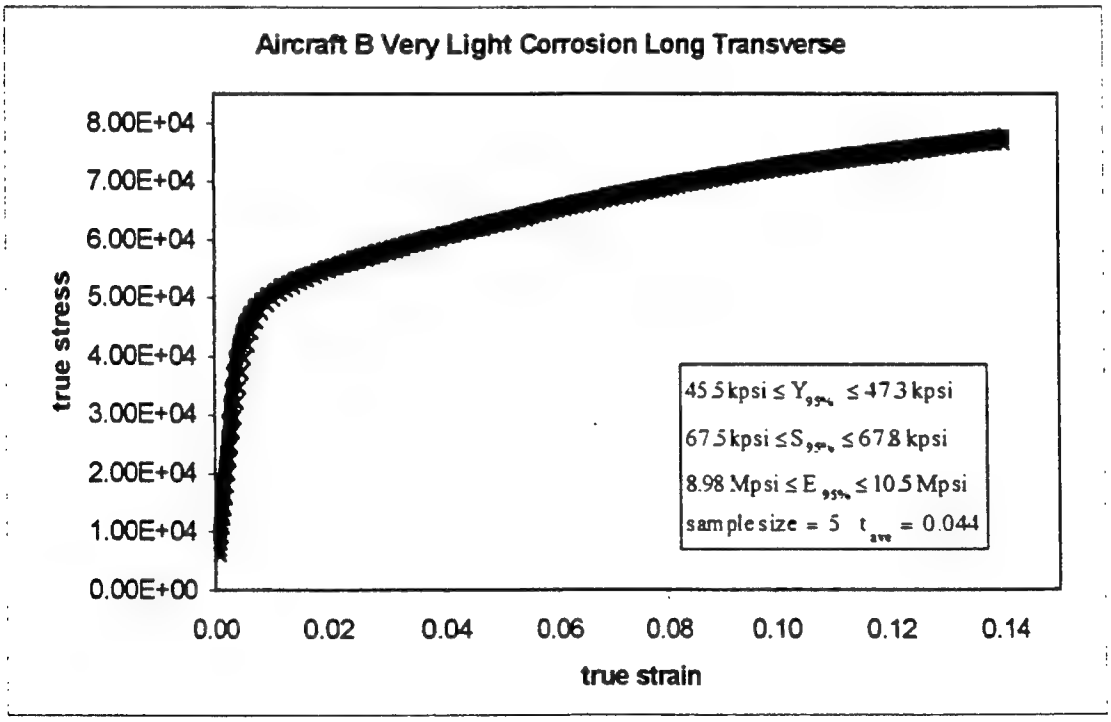


Figure 17. Light Corrosion Condition in Aircraft A, LT Direction

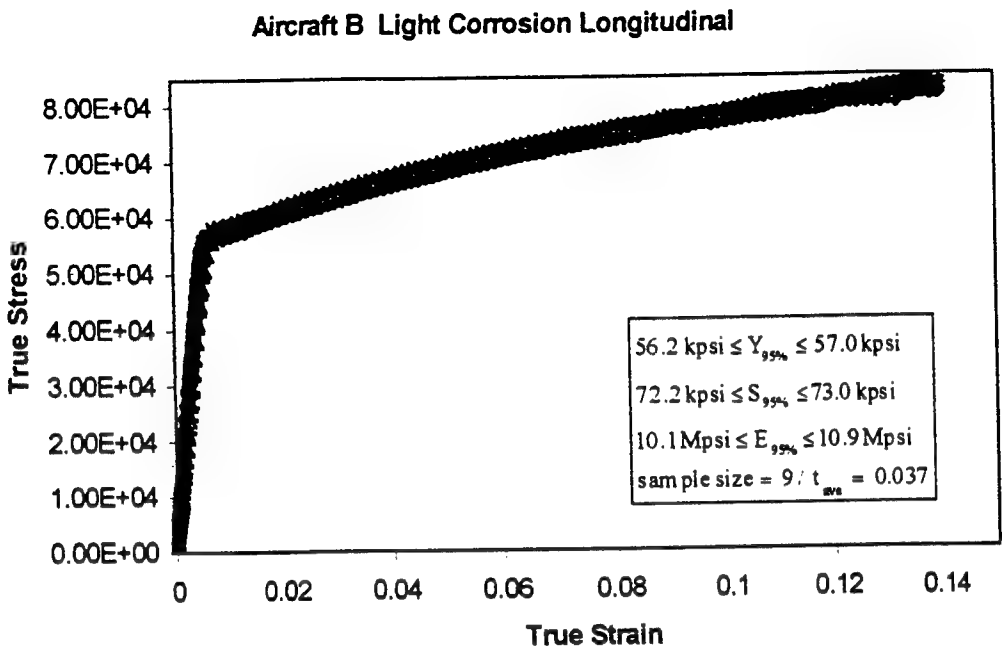


a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi; F_{ty} = 47 kpsi

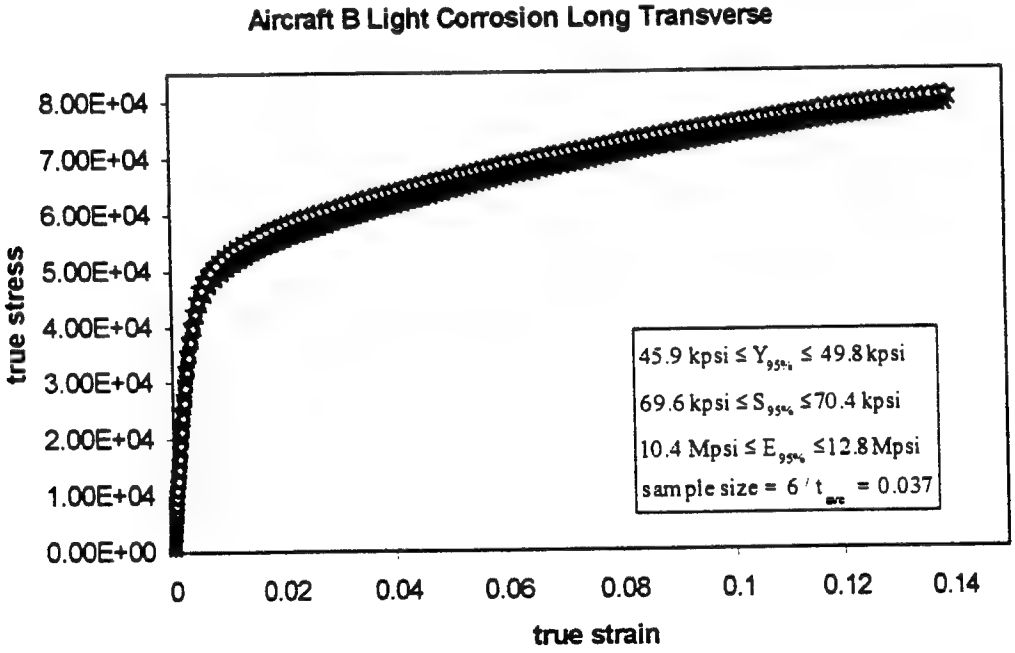


b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi; F_{ty} = 42 kpsi

Figure 18. Very Light Corrosion, Aircraft B, L and LT Directions



a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi; F_{ty} = 47 kpsi



b. Long Transverse: From MIL HDBK 5, F_{tu} = 63 kpsi; F_{ty} = 42 kpsi

Figure 19. Light Corrosion Aircraft B, L and LT Directions

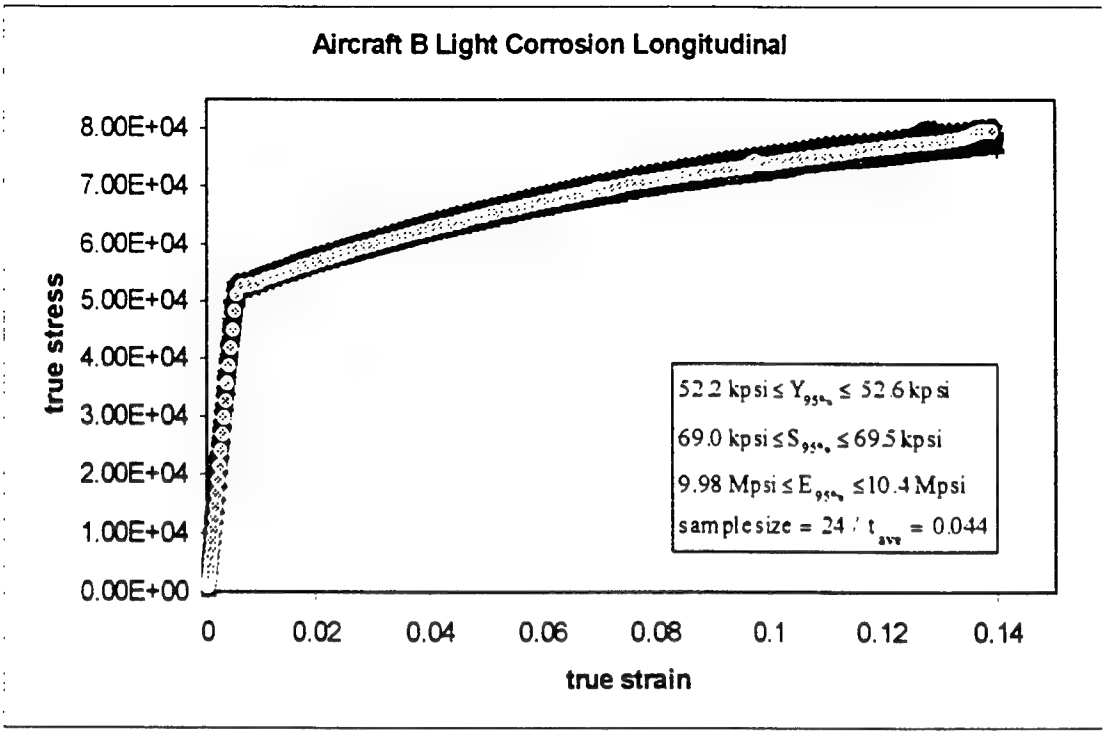


Figure 20. Light Corrosion Aircraft B, L Direction, Alternate Population

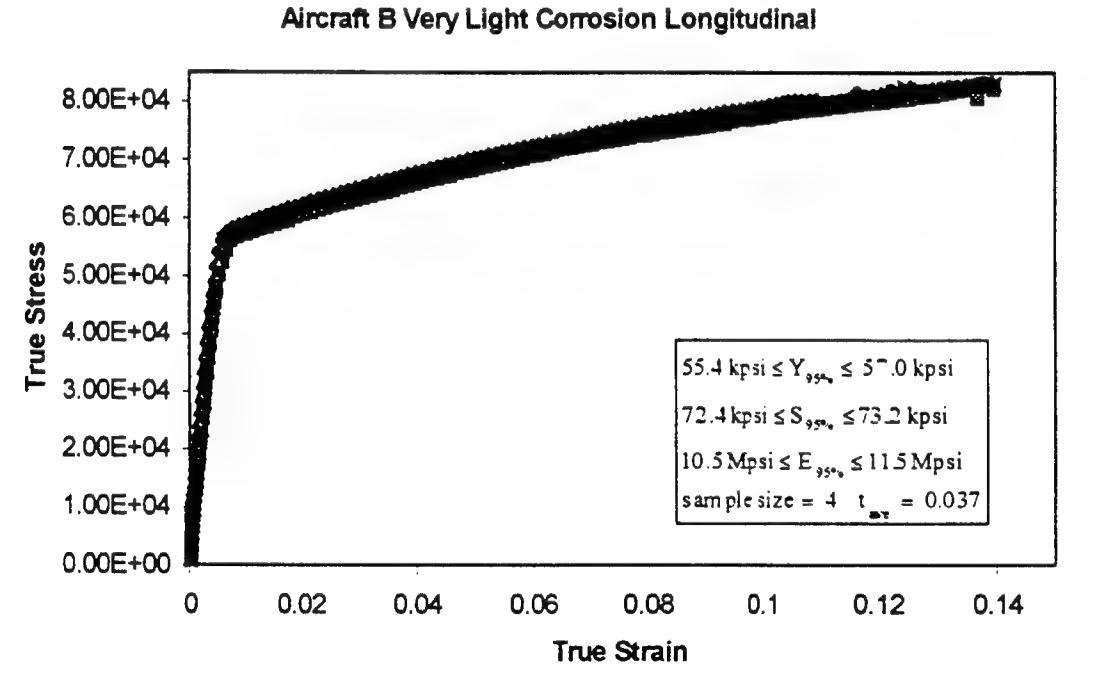
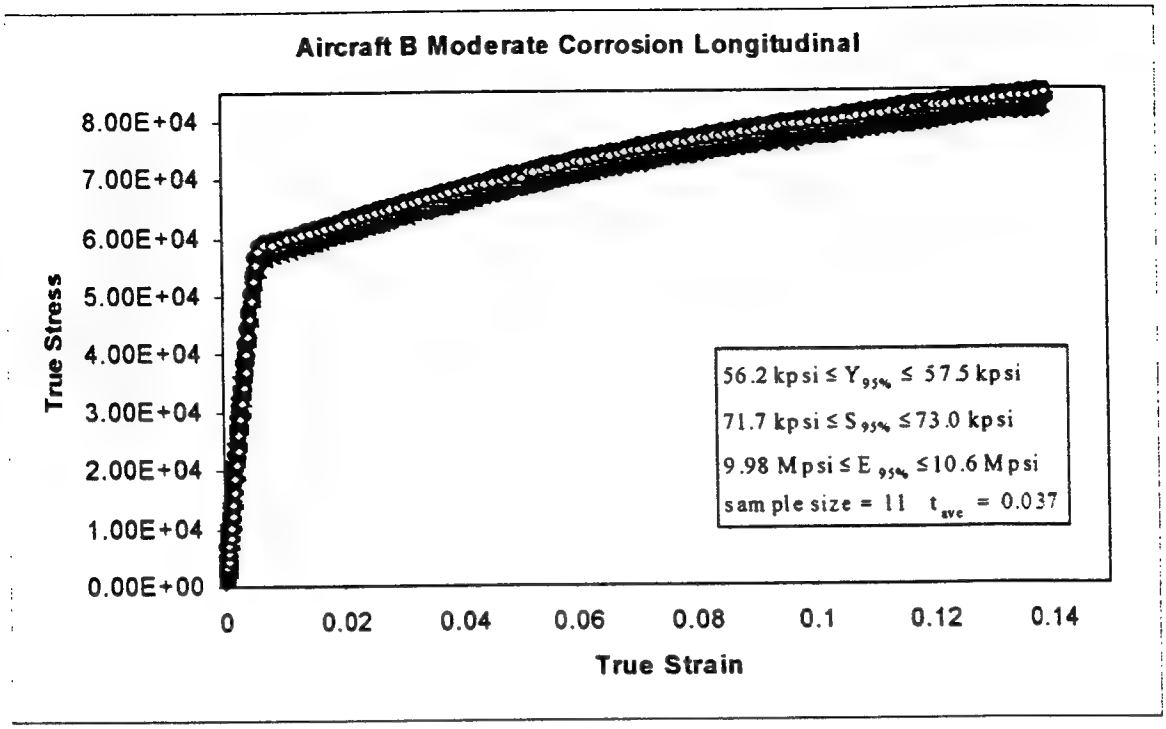
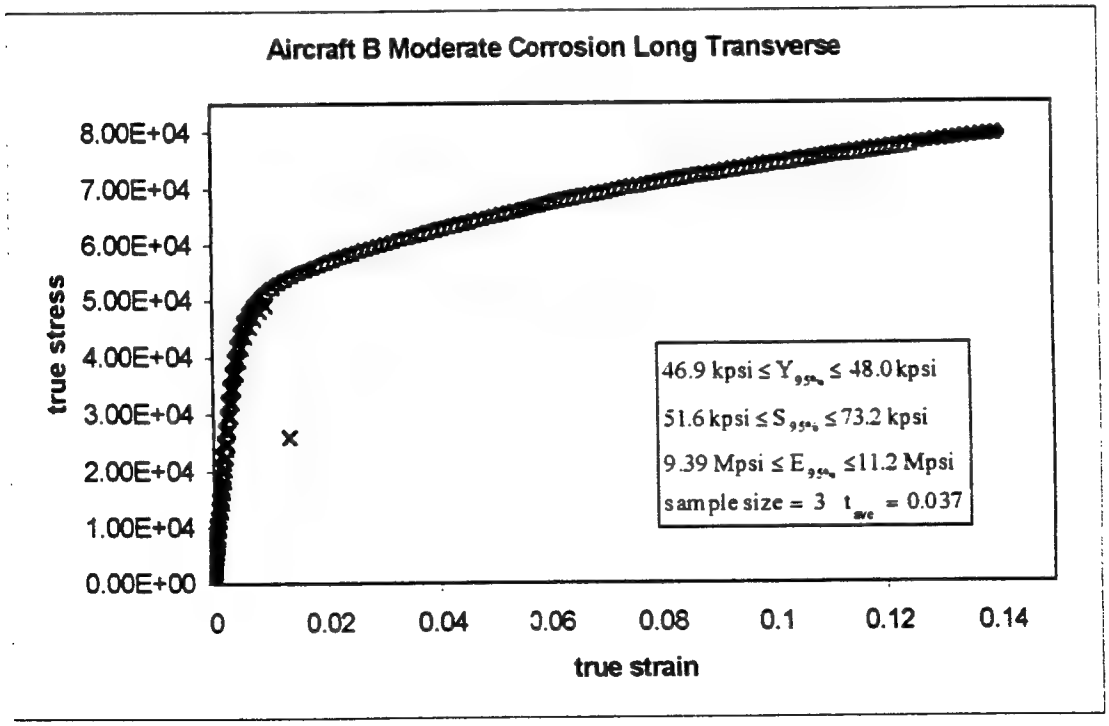


Figure 21. Very Light Corrosion Aircraft B, L Direction, Alternate Population



a. Longitudinal: From MIL HDBK 5, F_{tu} = 64 kpsi; F_{ty} = 47 kpsi



b. Long Transverse: From MIL HDBK 5. F_{tu} = 63 kpsi; F_{ty} = 42 kpsi

Figure 22. Moderate Corrosion Aircraft B, L and LT Directions

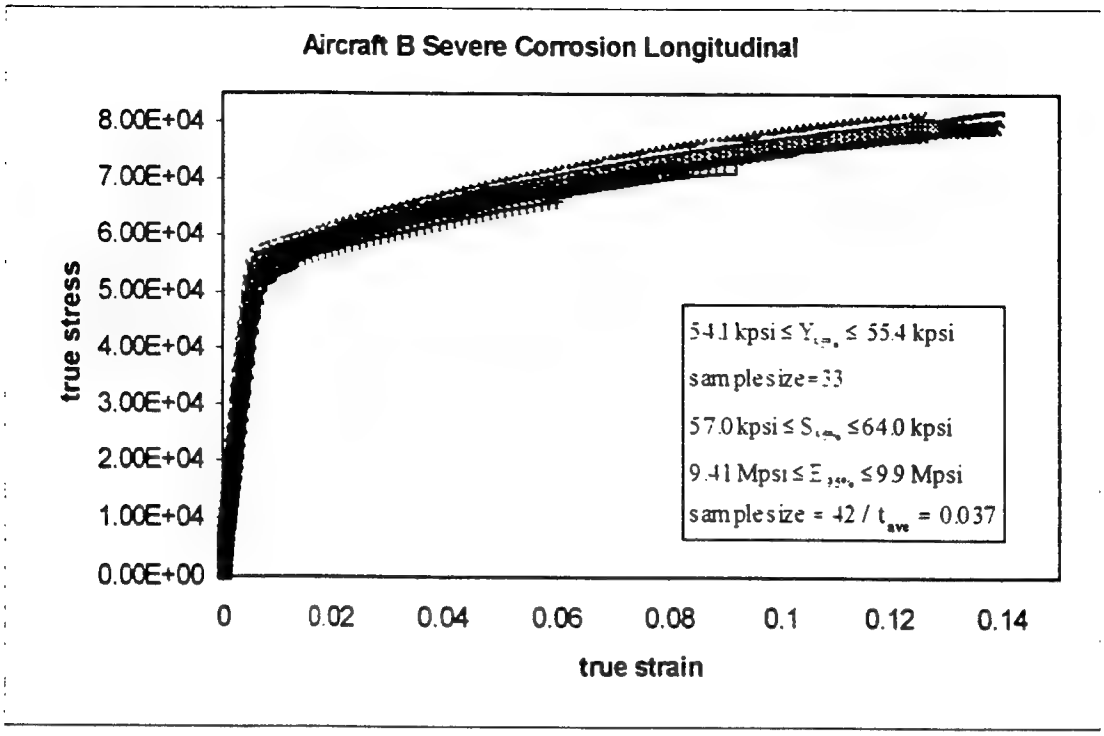


Figure 23. Severe Corrosion Aircraft B, L Direction

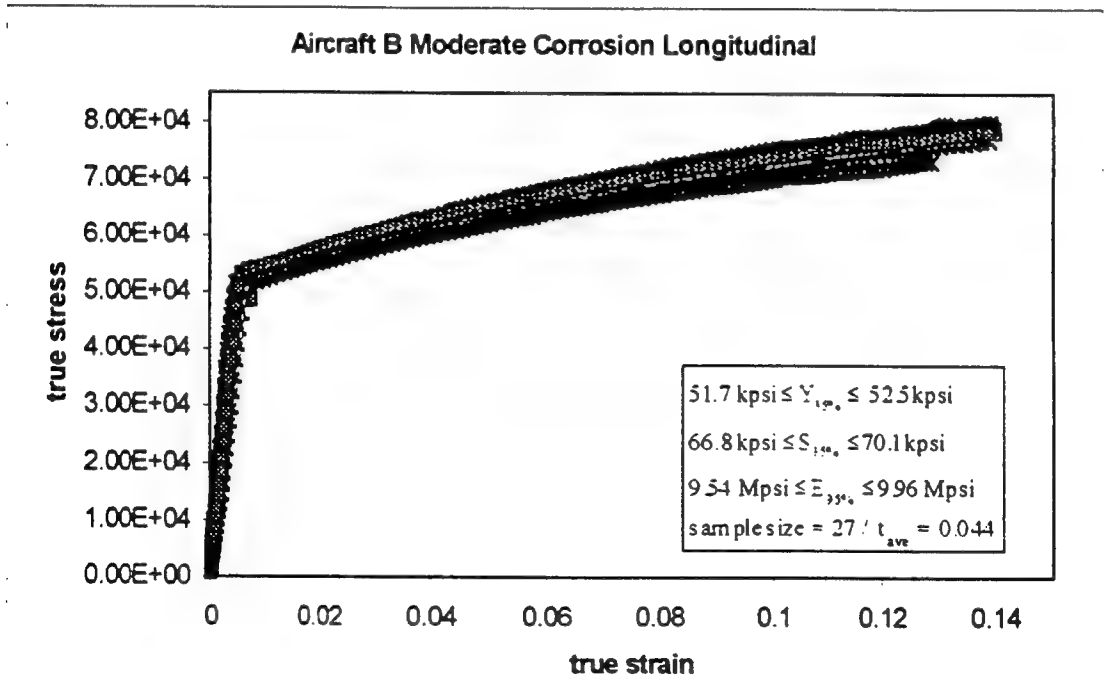


Figure 24. Moderate Corrosion Aircraft B, L Direction, Alternate Population

FURTHER DEVELOPMENT OF A SIMPLE, MULTIVERSION CONCURRENCY CONTROL PROTOCOL FOR INTERNET DATABASES

Devendra Kumar
Associate Professor
Department of Computer Sciences

The City College of New York
Convent Avenue at 138th Street
New York, NY 10031

Final Report for:
Summer Research Extension Program

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, DC
and
The City College of New York

January, 2000

FURTHER DEVELOPMENT OF A SIMPLE, MULTIVERSION CONCURRENCY CONTROL PROTOCOL FOR INTERNET DATABASES

Devendra Kumar
Associate Professor
Department of Computer Sciences
The City College of New York

Abstract

We present a concurrency control protocol that would be particularly suitable for large, distributed databases accessible via the internet. We also present its correctness proof. The main function of a concurrency control protocol is to make sure that the database remains consistent and coherent in spite of concurrent access to it by different users who are reading and updating it. Internet based distributed database systems have special requirements, e.g., having a large number of read-only transactions, and an expectation that such transactions should almost always be given a consistent set of data; letting a read-only transaction read an inconsistent set of data or only part of the data it needs, and later rolling it back is unreasonable— especially when it is an interactive transaction involving a human user, since it is difficult and annoying for a human user to "unlearn" what he read previously. Many traditional protocols do not satisfy this expectation. Also, a read-only transaction expects a fast response time, so it should not be normally delayed or rolled back simply due to the peculiarities of the concurrency control algorithm being employed. Unfortunately, most traditional protocols would do just that — delaying a read-only transaction due to unavailability of certain locks or otherwise rolling it back for a variety of reasons. Also, some of the read operations by an update transaction may be "useless" in the sense that they won't influence the computation of write values by the transaction. For example, a user may surf other parts of the database just to get a general idea of the contents. Traditional protocols do not take advantage of these reads being "useless" (i.e., insignificant) in optimizing performance; they simply assume that all reads are used in computing the write values in ways unknown to the system. Our concurrency control protocol handles all these issues effectively. Moreover, our protocol is conceptually simple, and easy to implement. It is quite flexible, allowing several variations that can be used for performance enhancements or user convenience. For example, a transaction is free to check if a new version of the database has been created after its previous read operations, and hence it may wish to discard the older version and may read again from the newer version. On the other hand, sometimes a read-only transaction may indeed prefer to read an older version of the database even though newer versions are available. In terms of common classification of concurrency control protocols, ours is a *multiversion*, *optimistic* protocol.

FURTHER DEVELOPMENT OF A SIMPLE, MULTIVERSION CONCURRENCY CONTROL PROTOCOL FOR INTERNET DATABASES

Devendra Kumar

1 Introduction

Many large organizations today maintain their databases over several sites that are geographically distributed. Examples include the military, various branches of the government, large corporations, etc. Many smaller organizations also maintain data over sites that are connected by a local area network. All these are examples of *distributed databases*, i.e., a database that is distributed over several sites.

Last few years have witnessed a growing popularity of the internet. We expect this trend to continue. We use the term *"internet databases"* to refer to distributed databases accessible to a large number of users over the internet. Examples of such applications include military information systems, healthcare databases, multinational business databases, etc. We expect the number of such databases to grow over the coming years, for a variety of reasons. First, most large organizations seem to have accepted the internet technology as an important technological tool to conduct their business, and are therefore providing internet access to their databases to their employees, customers, and other interested parties. Moreover, a growing population across the globe is finding the internet, and computers in general, to be interesting and useful. This encourages more people to access a given database. We are interested in the design of such databases.

In a distributed database, the data is distributed over several sites. Typically, different sites are geographically widely distributed, but sometimes they may also be part of a local area network. Different transactions, originating at various sites, operate on this data concurrently. The operations are to read data from the database, or to update it. If left uncontrolled, such concurrent executions of the transactions may leave the database in an inconsistent state or may lead to a situation that various data read by a user might be mutually inconsistent. The database is assumed to be in a consistent state initially. (Informally, consistent state means that the data is coherent and meaningful and is ready to be used by new users; it is not the case that part of data is old and part is new or still not written yet, and therefore the data as it exists does not make sense together.) Also, it is assumed that each transaction satisfies the following property: starting from any consistent state, if the transaction is executed alone, then the resulting database state at the end of this execution will also be consistent. Clearly, starting from the initial consistent state, any serial execution of one or more transactions will leave the database in a consistent state. However, if the transactions were to run concurrently, with arbitrary interleaving of their operations (such as reading or

writing values to variables in the database), the resulting state may not be consistent. There are additional problems caused by uncontrolled, concurrent execution of transactions. Therefore one major component of a distributed database management system is the *scheduler*, which uses a *concurrency control protocol* to avoid these problems. The main function of a concurrency control protocol is to make sure that the database remains consistent and coherent in spite of concurrent access to it by different users who are reading and updating it. Moreover, it ensures that each user's view of the database is also coherent.

Also, various kinds of failures may occur in the system, e.g., a transaction may explicitly ask to abort itself due to some undesirable conditions, or the operating system might crash, or the database management system may abort the transaction due to a deadlock, or the main memory may fail, or the disk may crash, etc. The *recovery manager* deals with such failure conditions; in particular it makes sure that the database remains consistent in spite of such failures. The recovery manager does its job in coordination with the scheduler.

Several concurrency control and recovery protocols for distributed databases have been designed and studied over last several decades; we refer the reader to any of several excellent works [5,9,17,3].

For a given application, performance of various concurrency control and recovery protocols may vary significantly. Several different factors might influence the choice of an appropriate approach for a given application, e.g., the degree of concurrency, the degree of conflict among transactions in terms of updating the same data items, etc.

Our main goal in this research is to develop a concurrency control protocol suitable for internet databases. In doing so, one approach would be to simply use one of the well known protocols for distributed databases. However, we believe that most of these protocols have certain shortcomings with respect to specific requirements of internet databases. Note that most of these protocols were designed before the advent of "the internet", i.e., the widespread use of internet technology among a large number of common users. In Section 2 below, we have listed several characteristics and requirements of internet databases that make them different from the more traditional distributed databases. But here let us briefly look at some of these requirements.

One requirement of internet databases is having a large number of read-only transactions, and an expectation that such transactions should almost always be given a consistent set of data; letting a read-only transaction read an inconsistent set of data or only part of the data it needs, and later rolling it back, is unreasonable— especially when it is an interactive transaction involving a human user. It is difficult and annoying for a human user to "unlearn" what he read previously. Many traditional protocols do not satisfy this expectation. Also, a read-only transaction expects a fast response time, so it should not be normally delayed or rolled back simply due to the peculiarities of the concurrency control algorithm being employed.

Unfortunately, most traditional protocols would do just that — delaying a read-only transaction due to unavailability of certain locks or otherwise rolling it back for a variety of reasons. Also, some of the read operations by an update transaction may be "useless" in the sense that they won't influence the computation of write values by the transaction. For example, a user may surf other parts of the database just to get a general idea of the contents before he starts updating the database. Traditional protocols do not take advantage of these reads being "useless" (i.e., insignificant) in optimizing performance; they simply assume that all reads are used in computing the write values in ways unknown to the system.

The traditional way to develop distributed protocols is first to develop a protocol that assumes a centralized scheduler, i.e., the scheduler is assumed to be on one site. Focusing on the central scheduler in the initial design allows us to look at the fundamental characteristics of the protocol. Similar approach is commonly taken in the literature, e.g., in [5,9,17,3], a protocol is discussed in the centralized case first, and is later extended to the distributed case. We have taken this approach in this research and have developed a protocol based on a centralized scheduler that we present in Section 3. We call this protocol the Internet Database Protocol (IDP).

Our concurrency control protocol handles all the special requirements of internet databases, as listed in Section 2, effectively. Moreover, our protocol is conceptually simple, and easy to implement. It is quite flexible, allowing several variations that can be used for performance enhancements or user convenience. For example, a transaction is free to check if a new version of the database has been created after its previous read operations, and hence it may wish to discard the older version and may read again from the newer version. On the other hand, sometimes a read-only transaction may indeed prefer to read an older version of the database even though newer versions are available.

In terms of common classification of concurrency control protocols, our algorithm is essentially an optimistic, multiversion protocol [5,17] with one important difference — *we maintain version numbers for the database as a whole, not for the individual items of the database.* Each version of the database is a consistent version, and hence can be read by a transaction without causing a retrieval inconsistency [5,3]. This simple view results in tremendous benefits in terms of handling special requirements of internet databases, and the resulting algorithm is also very simple and flexible. In traditional multiversion protocols, the latest version numbers of different data items is different; so at any given moment, it is difficult to take a snapshot of one consistent, coherent database while other users are updating it. In our case, one simply has to read all data items corresponding to the same version number.

In Section 4, we discuss correctness of IDP. In Section 5 we comment on its performance. Our algorithm is quite simple and flexible. In Section 6, we discuss a few simple variations of IDP.

Since ours is a *multiversion* protocol, in Section 7 we compare our protocol with the multiversion

protocol discussed in [17] in terms of performance, flexibility, and simplicity, and show some examples where our protocol does better in terms of the requirements of internet databases.

2 General Characteristics and Requirements of Internet Databases

Here we consider a large, distributed database that is accessible to a large number of users via the internet, intranet, or similar technology. As the number, size, and capabilities of these systems grow, a set of common characteristics and common requirements regarding their usage and system performance expectations seem to be emerging that are relevant to the database designer. In this Section, we list some of the main such characteristics and requirements that are over and above those commonly accepted for the more traditional distributed database systems. In other words, we do not repeat here what is generally known for common distributed database environments. Of course, exact usage patterns and requirements among various internet databases will vary; therefore, for a specific system, some of these observations might not apply, or there may be additional requirements.

1. The system will have a large number of users, many of whom are simply read-only users, i.e., they do not have an interest in, or the authorization to, update the database. These users will generate a large number of concurrent, read-only transactions. A read-only transaction is one which does not update the database. For example, a test engineer may wish to find out what test equipment and test procedures are needed to test a particular aircraft. In contrast, an update transaction is one that updates the database; it may or may not read data from the database.

2. A read-only transaction will not always be able to declare at its beginning that it is indeed a read-only transaction. For example, a user may read some data with the intention of updating the database, may even write some updates in his own workspace, but later may decide not to update the database after all. Thus it turns out to be a read-only transaction in the end. The database designer should not insist that a read-only transaction declares itself to be as such at its beginning.

3. Most read-only transactions expect the system to give them a consistent set of data in the first instance. (We say that the data read by a transaction is consistent if it came from a consistent version of the database that existed at some point in time.) It is generally not acceptable for the system to first let them read inconsistent data and then later tell them that the data was inconsistent and therefore they need to start over again. Several protocols do not satisfy this requirement. For example, in the multiversion protocol discussed in [17], an update transaction T1 may be aborted and as a result a read-only transaction T2 that previously read data written by T1, may also have to be aborted.

There are several reasons for the above requirement in the context of an internet database. First,

it is sometimes difficult for a human user to unlearn what he has learnt from the data he has already read. It also causes delays which may be annoying to the human user. Moreover, note that in such an environment, external writes [5] are common. A user may read the inconsistent data and may use it in some other application, fax it to somebody, or take decisions based on it— all this has to be reversed when the transaction is rolled back. Sometimes an external write may be irreversible; in such instances the user has to delay the external write until he is guaranteed by the system that the data he read is consistent (in general, this is true when the transaction is finally committed by the system; in special cases a concurrency control protocol may guarantee that all data that is being read by the user will be consistent). Indeed, in some applications, a system that rolls back a read-only transaction may be considered incorrect, rather than just very slow. However, we will not take this extreme view in our discussions in this paper.

4. It is also generally not acceptable to give the user a consistent set of data, but then ask him to roll back and start over again before he asks for further data (the data to be read after the roll back would normally be inconsistent with the one read before the roll back). For example, a locking protocol may result in a deadlock involving such a transaction and hence it may have to be aborted. The reasons for this requirement are similar to above.

5. Most read-only transactions expect a fast response time since they involve an interactive human user. There are several implications of this: (a) The concurrency control algorithm should not unduly delay such transactions. For example, in the two-phase locking protocol [5], a read operation on a data item requires getting a lock on the item, and sometimes this may cause a large delay if the lock has been granted to an update transaction. (b) Similarly, rolling back read-only transactions should be kept to a minimum since that causes large delays. Examples of rollbacks of read-only transactions have been mentioned above. Now suppose a user has read some data and has spent several minutes looking it over, and then that transaction gets rolled back. Then these wasted minutes have to be included in determining the response time. Similarly, when a user has to delay an external write because the data is not yet guaranteed to be consistent, this delay also becomes part of the response time. In general, we define the response time to be the time interval from the moment the transaction issues a read operation first time (over many lifetimes due to rollbacks) and the moment it knows that the data it has received, or has started receiving, is guaranteed to be consistent. Thus rolling back a transaction increases its response time.

6. Many users will generally accept getting a somewhat older version of the database. For example, if a user is trying to read test equipment data regarding an aircraft, he will generally be comfortable reading data that is just a few weeks old and is not necessarily the most up to date version of the data. This is more true for non-critical data such as finding the phone number of an individual. The database design should try to take advantage of this in its attempts to minimize response time or to increase system throughput.

7. Some update transactions also involve interactive, human users and therefore they also expect fast

response times. Comments above regarding read-only transactions involving humans also apply here.

8. Many transactions (both read-only and update transactions) are of long duration since they involve an interacting, human user. The system should take special care to make sure that they do not affect system performance significantly. For example, in a locking based protocol, if such a transaction is holding locks, it may delay other transactions significantly.

9. Many update transactions have "useless reads" in them. For example, a user writing a report might wish to see a report written by a colleague to get a feel for the general contents or structure— such a read does not affect the actual contents of his own report. Similarly, the user might simply want to surf the database before starting his writing work. Such reads are "useless" reads; they violate a common assumption made in traditional concurrency control theory that every write value is an unknown function of every data previously read by the transaction. With internet databases, this assumption is no longer valid. The database system should try to take advantage of this additional information to improve performance. In our algorithm, we allow a transaction to effectively tell the system which of its reads were actually useful reads, and thus the useless reads have no effect on the workings of the concurrency control algorithm.

3 Our Concurrency Control Algorithm

Conceptually, the database goes through a sequence of versions, as a result of updates by transitions. The initial version number of the database is 1. With time, the version number changes through the sequence of numbers 1,2,3,.... Each version is guaranteed to be one consistent version of the entire database. How does the database go from a given version I to the next version I+1? Normally this happens when, effectively, some transition T reads version I, updates the database to version I+1, and the system commits to this update. At other times a transaction reads contents of version I, then gets an authorization to update the database, but aborts before completing the update— in this case contents of version I+1 remain the same as version I. As is commonly assumed in concurrency control algorithms, when a transaction reads from a consistent database and subsequently completes all its updates in isolation from other transactions, the resulting database is also consistent. Thus we see that each version represents a consistent database.

Note that our use of version numbers is quite different from that in other multiversion protocols— in our case, version numbers are assigned to the database as a whole, and not to individual data items. Sometimes we might use a phrase like "version VN of data item X"— this only means "the data item X in version VN of the database".

We are flexible as to how data is partitioned across the network. A given site might store only some of the data items. Also, a given data item may be replicated at many sites. For the purposes of our concurrency

control algorithm, these variations are not important. As is the common convention, these data items can be a relation, or part of a relation, etc.

Suppose site S stores a data item X. Then along with a value of X, it will also store the corresponding version number.

There are two main bodies of information maintained in the system (at the scheduler or the sites):

(a) information as to which site has, in its storage, which versions of which data items, and

(b) information about what versions of the database exist anywhere in the system (without regard to their location), and which data items changed in which version of the database.

These two bodies of information are discussed in Sections 3.1 and 3.2. Note that this is only a conceptual view of what information is being maintained in the system regarding data stored at the sites, and changes in different versions of the database. Other data structures to represent this information are clearly possible; and this issue is not important for the concurrency control algorithm.

3.1 Information about Location of Stored Data

Each site and the scheduler will maintain information that tells what versions of what data items have been actually stored at this site. This information can be updated from time to time via communications among the sites and the scheduler; how that is done is not important to our algorithm. This information is useful when a transaction has decided that it wants to read a data item X in a version number VN, and needs to find out which site can provide this data. This information is also useful when a data item is to be updated.

3.2 Maintenance of Information on Version Changes in the Global Database

The scheduler maintains global information about the changes to various data items as the versions of the database change (but not the actual values of the data items). Specifically, For each data item X, it maintains a linked list of database version numbers which have been committed to by the system, and which resulted from an update to this item (along with updates to possibly some other items as well). In other words, these are the database versions where X has a new, updated value (occasionally, an updated value may turn out to be the same as the previous value, but that is not important. So to simplify the discussion, in our explanations we will assume that each updated value is distinct from all previous values. However, note that this is not assumed by the algorithm itself). This list, called the VLIST (Version LIST) is in sorted

order by version numbers; the highest version number being at the head of the list. Thus if two consecutive nodes on this list are VN1 and VN2, with VN1>VN2, then that means that the value of X has changed in versions VN1 and VN2; but note that in future, we might insert another node VN3 in this list where VN3 is in-between VN1 and VN2. (The reason for this is that the database version VN3 has been authorized by the scheduler, and X is planned to be updated in that version, but the recovery manager has not finally committed to that update yet; in fact, this update might get aborted due to failures, and therefore may not be inserted in the VLIST. This will become clearer shortly.) If the highest version number on the list for X is VN, then that means that the value of X has changed in version VN, and similar to VN3 above, a higher version than VN might have been authorized but has not been committed by the system yet.

The scheduler has a variable, named CV (Current Version), which is the highest version number such that no other version number VN≤CV will ever be inserted in the VLIST of any data item in the future. Therefore all the version change information up to version CV has already been included in VLISTs of all data items and no such node will ever be added in the future. This means that no new versions below CV+1 are currently planned, or will be planned in future, or will be committed in the future. Thus if two consecutive nodes on the VLIST for data item X are VN1 and VN2, with VN1>VN2 and CV≥VN1, then we are guaranteed that the value of X is same in versions VN2, VN2 +1, ..., VN1 -1 since no new node will be added in this range. Similarly, if the highest version number on the VLIST for X is VN and CV>VN, then that means that the value of X is same in versions VN, VN+1, ..., CV. So if a transaction has read X in version VN, it may assume that version CV will have the same value.

Similar to the above VLISTs and CV at the scheduler, each site also maintains its own copy of the VLISTS and CV; but they might not be as up to date as the scheduler. This is just a snapshot of the VLISTs and CV of the scheduler, but perhaps a bit older. The purpose of having this information at a site is to provide quick information, without communication with the scheduler or the other sites, to a read-only transaction which is willing to accept a somewhat older version of the database. If it is important to get the latest information, e.g., in case of an update transaction, it will have to communicate with the scheduler. We make the following observations about these variables at a site:

(a) The scheduler maintains VLISTs for all data items in the global database whereas a site may only be interested in some of the items— typically the data items it stores and possibly a few others that transactions originating at this site are typically interested in. Therefore the site might not have VLISTs of all data items.

(b) The exact values of these variables will not be an exact copy of the values at the scheduler because of delays involved in information exchange.

(c) CV and VLISTs at a site have information regarding version changes in the global database; and not information regarding which versions of which data items have been actually stored at the particular site (this is a different kind of information, and was discussed earlier in Section 3.1). For example, at site S

it is possible to have VLISTs up to version 10, and CV=10, but the actual data values stored at this site are only up to version 5— the later versions of the data have not yet arrived at this site.

(d) The values of CV and VLISTs, at a given site, must be mutually consistent with respect to their meanings discussed earlier for the scheduler. For example, we should not have a case where, at the scheduler, VLISTs have some nodes with version number 10, and CV=10; and at site S we have CV=10 but the VLISTs do not have any nodes with version numbers 10 because that information has not yet arrived from the scheduler. The information at site S is incorrect because this says that no data items were updated in version 10 (recall that no new nodes will be inserted with version number less than or equal to CV).

(e) The value of CV at any one site is a local value. Different sites might have a different value of CV.

3.3 Behavior of a Transaction and the Transaction Manager

Some of the actions described below are taken by the transaction itself, and some are carried out by the transaction manager on behalf of the transaction. We will not make the distinction between the two.

Before a transaction issues its first read operation, it needs to decide from which version of the database it is going to read various data items. There are several possibilities here. In general, a transaction can find out the CV value at the scheduler or at a particular site by sending a GETVN query to it. However, note that only the scheduler has the latest value of CV. If a transaction needs the most recent data, then it must send the GETVN query to the scheduler. An update transaction would typically need the most recent data. A read-only transaction might prefer an older version, e.g., if that version is actually stored at that site or it had read that version previously.

When a transaction issues a read operation, it specifies which version of the database it wants to read those items from. As long as all the reads have the same version number, the system guarantees that all the values read will correspond to one consistent version of the database. So normally a transaction would use the same version number in all its read operations.

When a transaction issues a write operation, the updates are made in the workspace of the transaction, not the database itself. This is similar to the validation protocol discussed in [17].

When a read-only transaction, i.e., one which does not wish to update the database, executes its commit operation, it simply terminates, without having to communicate with the scheduler. Unlike the validation protocol, it does not need to check if its reads were consistent. If it used the same version number, then automatically they were consistent.

When an update transaction executes its commit statement, a validation check needs to be performed at the scheduler to check if the updates by the transaction are to be allowed. To this end, the transaction sends an UP (Update Permission) query to the scheduler with the following parameters:

X1, X2, ... : list of data items that it read and that were used in computing its update values (thus "useless" reads need not be mentioned in this list). This is also called the read set of the transaction.

VN= the version from which it read the above data items (all of them must have been read in the same version)

Y1, Y2, ... : list of data items that it wants to write to. This is also called the write set of the transaction.

The scheduler tests whether this update can be allowed, i.e., it will maintain database consistency; we will discuss this test shortly in Section 3.4. If the test fails, the scheduler sends a message DENIED to the transaction, indicating that this update will not be allowed. At this point the transaction will abort itself, and then restart. On the other hand, if the test passes, the scheduler finds a new version number (say VN2) for the database which would potentially result from this update; we will discuss in Section 3.4 how this number is generated. Then, the scheduler will send a message OK(VN2) to the transaction. On receiving this message, the transaction will update the database. Note that there is no guarantee that the updates will actually be committed on the stable storage— because there may be hardware or software failures. In case a failure occurs, the transaction will abort and restart. In case the update gets committed by the system, the transaction will gracefully terminate.

3.4 Behavior of the Scheduler

The UP queries arriving at the scheduler are kept in a FIFO queue and are processed in serial order. Notice that at any moment, the scheduler may have given update permission to several transactions. It needs to maintain information about them in order to facilitate future updates to its CV and VLISTs. Also, this information is needed to validate future update requests. So the scheduler maintains a list (say, PU— Pending Updates) of records, each record corresponding to one update permission. Contents of such a record are: Version number assigned to this update, Transaction id, The write set.

In addition, a variable PV (Pending Version) holds the highest version number assigned to any transaction— whether it has already committed, aborted, or is still pending.

Below is the test to certify a validation request UP(R, VN, W) where R= the read set and W= the write set:

(a) For each item X in R: the VLIST of X has no node with version number greater than VN. In other words, none of the data items in R has been updated (i.e., with commitment by the system) with a version greater than VN, and

(b) For each record in the PU list: the write set of the record has no common element with R.

Essentially, this ensures that the items read by the transaction have not been updated in more recent versions, and will not be updated by any transaction in the PU list.

If the above test fails, the scheduler sends a DENIED message to the transaction.

If the above test passes, then PV is updated to PV+1, and this new PV value becomes the new version number assigned to this update. A new record corresponding to this update is inserted in PU, and the transaction is sent an OK(PV) message.

Now let us consider what the scheduler does when an update in the PU list is finally committed or aborted by the recovery manager.

In Case of an Abort by the system:

```
Let VN be the version number in the corresponding record in PU list;
delete the above record from the PU list;
if VN=CV+1 then
begin
        CV := CV+1;
        continue := true;
        while (CV < PV and continue) do
        begin
                if there is a record in the PU list containing version number
                        CV+1 then continue := false
                else CV := CV +1
        end;
end;
```

In Case of a Commit by the system:

```
Let VN be the version number in the corresponding record in PU list;
Let W be the write set in the corresponding record in PU list;
for every data item X in W do
        in the VLIST for data item X, insert a node with value VN;
delete the above record from the PU list;
if VN=CV+1 then
begin
        CV := CV+1;
        continue := true;
        while (CV < PV and continue) do
        begin
                if there is a record in the PU list containing version number
                        CV+1 then continue := false
                else CV := CV +1
        end;
end;
```

When the system commits to an update, if the version number VN of this update is not CV+1, then note that CV is not advanced to VN. Doing so would be semantically incorrect in terms of the meaning of the CV, since the status of some earlier versions is still unknown; and it would cause inconsistency retrieval problem, since later an earlier version may get committed. For example, suppose before a particular commitment, CV=5 and the update is for version 10, and the write set has X and Y in it. Suppose some other data item Z, not in this write set, has the highest version number equal to 7 in its VLIST and there is a record in PU with Z in its write set and version number being equal to 8. After the commitment of version 10, if we move CV to 10 then semantically we are declaring that the value of Z does not change in versions 8, 9, and 10. This is semantically incorrect since we don't know if version 8 will get committed or not. Moreover, if we changed CV to 10, then later some transaction T might find that CV is 10, and it may issue a read of X and Z in version 10 (the system will supply the value of Z from version 7, since CV = 10 and therefore Z is supposed to be the same in version 10). In case version 8 gets committed, these values of X and Z do not represent values from a consistent database.

4 Correctness of IDP

4.1 Safety Properties

Referring to [29], safety properties assert that *"the program does not do something bad"*. A safety property is a predicate on the states of the system, claimed to be true either when the *system computation is terminated* (defined next) or for every reachable state of the system.

In our case safety means that the protocol is serializable. In other words, we need to show that irrespective of whether the pending updates in the PU list get committed or aborted, each version of the database is consistent, and is the result of a serial execution of zero or more transactions.

Note that the VLISTs together define the permanent (committed) database versions at any time. Suppose in the VLISTs, we have two versions VN1 and VN2, with VN1<VN2 and there is no version VN3 in VLISTs such that VN1<VN3<VN2. For conceptual simplicity, we will say that the database in any such version VN3 is same as the database in version VN1. Of course, there may be a different database in version VN3 in the PU list which may or may not get committed.

Theorem 1: Within the committed versions of the database, each version VN1 is a result of a transaction that reads from version (VN-1).

Proof: If there is no VN1 version, then this transaction is the null transaction. Otherwise, consider the moment when an entry is made in the PU list. Suppose this is from a transaction T with read-set=X, write-set=Y, read-version=VN1, (i.e., the version in which data items were read), and write-version=VN2 (i.e., the version in which the data items are to be written). Since the scheduler has authorized this entry, VN1 must already be in the committed database and the current PU does not have an entry whose write-set conflicts with X. Therefore even if the other entries get committed earlier, they will not interfere with this transaction since its read-set values are not updated by them.

From Theorem 1, it follows that the schedule is serializable.

4.2 Liveness Properties

Liveness properties in general assert that *"the program does eventually do something good"* [29]. More specifically, if a read-only transaction wants to read something, it will do so within a finite time. Similarly among the update transactions, at least one should be able to make progress.

Theorem 2: Suppose a read only transaction wants to read data. It will be able to do so within a finite time.

Proof: Note that a read-only transaction always has the option of reading the database at version CV or earlier. So it can always read it. There are no locks to obtain.

Theorem 3: Suppose one or more update transactions want to update data. Then within a finite time at least one will be able to make progress.

Proof: The only situation where this is violated is when the PU list remain empty forever, i.e., all new UP requests are denied. But if the PU list is empty, then the value of CV will get updated to be the last version number in the VLISTs; and hence next time when a transaction reads data at current CV version, and subsequently makes an UP request, it will be granted.

5 Some Performance Related Observations on IDP

1. A read-only transaction is never rolled back by the *scheduler*. (It may be rolled back by the *recovery manager* due to hardware or software failure, but that is unavoidable.)
2. A read-only transaction is never delayed while waiting for an event to take place at other transactions. For example, it does not have to wait for certain locks to become available, etc. The only delays are communication delays and computation delays when a processor (executing this transaction or the scheduler) is executing some other program.
3. To get above benefits, a transaction does not even have to know at its beginning that it is going to be a read-only transaction. It does not have to declare to the scheduler that it is a read-only transaction. For example, it might read some data, and based on data values it may decide that it does not want to do any updates.

6 Simple Variations of the Algorithm

Several variations of the algorithm are possible which may improve user convenience, performance improvements etc. Here we suggest some of the possibilities.

6.1 Version Notification Feature

Consider the following situation. An update transaction T has read items X and Y in version 5, and is continuing with more read operations or computing values for write operations. It is quite possible that there is a newer version in the database and X has changed in this version. Therefore the present work of T will go waste since it will not get validated later. It would be useful to inform T of this version change so that it can restart or selectively re-read the items that have changed. Even if X has changed in a pending update (i.e., the update is recorded on the PU list but is not committed or aborted by the system yet), it would be useful for T to know about it.

Therefore, in order to improve performance of the algorithm in such situations, we include the following feature in the algorithm. A transaction T may send a request NOTIFY(VN, RLIST) to the scheduler. VN is a version number and RLIST is a list of data items. If the VLISTs of these items or the PU list indicates that any of these items have changed, or may change if one of the pending updates gets committed, in any version beyond version VN, then the scheduler sends a VCHANGE(VN1) message to T. VN1 here is the highest version number known to the scheduler where one of the items would change. In case there is no such known change yet, the scheduler will keep the NOTIFY message in a list and will process it again every time a new record is inserted in the PU list.

When the transaction T receives the VCHANGE message, T may further interrogate the scheduler to find out exactly what has changed. We skip details of such an interrogation process here. Based on this, the transaction may roll over, or selectively re-read the affected items.

Typically a read-only transaction will not be using this feature; this is mainly useful to an update transaction. The NOTIFY message may be sent right after finding the current version number through the GETVN query, or it may be sent after the transaction has made some progress in its computation so that it has a better idea of what the set of useful reads may be. Of course, if it does not know the exact set of useful read, it may simply send a larger set as the parameter RLIST in the NOTIFY message. Indeed, it may send this message several times during its computation after getting replies to the previous ones.

6.2 Other Variations

1. Earlier we said that in the UP(read set, VN, write set), all the data items in the read set must have been read in the same version, namely VN. However, this need not be strictly true. For example, it is possible that the transaction T reads X from version 5, then finds that the version number has changed to 7 but X has remained the same, then it reads Y from version 7, and later it submits an UP query with X and Y in its read set and VN being equal to 7. This should be acceptable since the value of X is same in versions 5 and 7.
2. Note that we allow a transaction to read data from different versions. This may be useful to some read-only transactions. For example, such a transaction might want to read the latest available version of certain data items, even if some other items that it read are from a much earlier version. Of course, this does not guarantee that all the data read by the transaction will be mutually consistent. However, the protocol will allow this without causing any rollbacks.

7 Comparison of Our Algorithm with a Multiversion Algorithm

Now we compare our algorithm, i.e., the Internet Database Protocol (IDP) , with the multiversion protocol (say MVP) discussed in [17]. MVP is also based on multiple versions, but the main difference is that the version numbers are assigned to individual data items, not the database as a whole. So there is no simple way to take one consistent snapshot of the database. In our protocol, each version number represents one consistent version of the database. We first briefly review MVP.

7.1 Brief Review of MVP

In MVP, each transaction T is assigned a unique timestamp, denoted by TS(T). Intuitively, the protocol attempts to schedule the operations of the transactions in such a way that the net effect is as if they executed serially, in their timestamp order. One typical way to generate the timestamp is to use the value of the system clock when the transaction begins its execution. In a distributed system additional care is taken to make sure that the transactions running at different machines will still be guaranteed to have different timestamps.

Each data item in the database has several versions. As in IDP, different values of a data item are stored along with their version numbers. In MVP, the version number of any version of a data item X is defined to be the timestamp of the transaction that created that version of that data item, by executing a write operation. This is also called the write-timestamp (WTS) of that version of X. With each version of X, we also keep another timestamp called the read-timestamp (RTS) of that version of X. This is the highest

timestamp of any transaction that read or wrote this version of X (of course, only one transaction could have written this version).

We use the term "latest" version of X to mean the version of X with the highest value of WTS.

In MVP, when a transaction wishes to read or write an item X, it issues a read(X) or write(X) operation. Unlike IDP, the operation does not specify which version is to be read or written. Also, unlike IDP, the operation goes to the scheduler which determines which version of the item is to be read or written, and whether the operation is permitted. In case of a write, the new value and its timestamps are written immediately to the database; unlike the case with IDP where the values are written in the local workspace of the transaction and written to the database only at the end of its code.

Behavior of the scheduler when it receives a read(X) request from transaction T:

Let X1 be the version of X whose WTS is the largest among all versions of X that
have WTS<TS(T);
if RTS(X1)<TS(T) then RTS(X1) := TS(T);
Permit T to read the version X1;

Behavior of the scheduler when it receives a write(X) request from transaction T:

Let X1 be the version of X whose WTS is the largest among all versions of X that
have WTS<TS(T);
if TS(T) < RTS(X1) then roll back T and any other transactions that read a
value written by T or by any other transaction being aborted hereby
else create a new version of X (say X2) with WTS(X2)=RTS(X2)=TS(T)

7.2 Performance Considerations

We consider several execution scenarios and compare the relative performance of the two algorithms for those scenarios.

Example 1: Using MVP, suppose the latest version of X is X1, WTS(X1)=RTS(X1)=100. Suppose TS(T1)=200 and TS(T2)=300. T2 is a read-only transaction. T1 wishes to read X, write X, and then commit. The following might happen:

T2 executes reads(X) and therefore reads X1. Now RTS(X1)=300.

T1 executes read(X) and therefore reads X1. Now RTS(X1)=300.

T1 executes write(X). Since TS(T1) < RTS(X1), T1 rollbacks.

If the same execution took place in IDP, there will be no rollback. This rollback is clearly unnecessary. Even if T2 were an update transaction, IDP will let T1 complete and commit while T2 is still computing. This speeds up the system throughput.

To put it more intuitive terms, MVP assigns a timestamp to the transactions according to when they start, and, informally speaking, hopes that they will execute their operations in the same order. It does not take into account the fact that a transaction that started later may be a short transaction that finishes up earlier. IDP commits to those transactions as they finish, irrespective of when they started.

Example 2: Consider two transactions T1 and T2 with TS(T1)<TS(T2). T2 simply wishes to read X,Y,Z. T1 wishes to write Y and write X. The following may happen in MVP:

T2 executes read(X).

T1 executes write(Y).

T2 executes read(Y).

T1 executes write(X)— this results in rollback. That causes T2 also to rollback since T2 has read data written by T1.

Here a read-only transaction is being rolled back. In IDP, no rollback will take place— both transactions will finish gracefully. This is because in IDP, T2 will read from a consistent version of the database that existed before these two transactions started.

7.3 Flexibility of the Algorithm

IDP and its simple variations provide a high degree of flexibility to the transactions. For example, a transaction need not decide at its beginning whether it is a read-only transaction or not; it may decide that after

reading some data. The protocol allows a transaction to re-read a data item in several different versions without making the algorithm itself more complex or having to rollback other transactions, etc. In MVP, when a transaction issues a read operation on a data item X, it has no control as to which version will be read; the concurrency control algorithm decides that for the transaction. In MVP, a transaction T1 cannot read data values written by transaction T2 if TS(T1)<TS(T2). This is undesirable in a situation where T2 has made substantial progress in its computation and is well ahead of T1, even though it started after T1. The notification feature provides ability of a transaction to keep up with any version during its own computation. Several variations mentioned in Section 4 further illustrate the flexibility of the algorithm. The algorithm itself can be modified in many ways, and new features can be added.

7.4 Simplicity of the Algorithm

IDP does not require generation of timestamps for the transactions. Only one timestamp is needed for individual data items, not two as in MVP (RTS and WTS). Conceptually also, the rules that the scheduler follows in dealing with write operations are much simpler.

In our protocol, storage of actual data values is independent of information about the versions of the database. Thus there is no need to synchronize various sites that hold multiple copies of a data item.

8 Concluding Remarks

We have presented a concurrency control protocol that is particularly suited for internet databases. Read-only transactions have been given special treatment and they are never aborted by the algorithm. The algorithm allows transactions to take long time durations in their computations without holding up the rest of the system. We believe the number of rollbacks would be far less in this algorithm than in the multiversion protocol, as illustrated by several examples in Section 7.2. The algorithm is fairly simple and flexible.

We have also presented a correctness proof of the algorithm. For easy readability, the proof is kept informal. A semiformal proof may be developed in future work.

Acknowledgments

We are thankful to Mr. David Kidd and Captain Janice Rodgers of San Antonio Air Logistic Center, Kelly Air Force Base for several discussions. We are thankful to Dr. Ravindran Krishnamurthy of HP Labs,

Palo Alto, California, for a long technical discussion on various parts of this research. Thanks are also due to Dr. Abraham Silberschatz of Lucent Technology, Murray Hill, New Jersey, for several discussions on synchronization issues related to this work. We also appreciate several discussions with Professor S. S. Iyengar of Louisiana State University on the synchronization aspects of the problem and approaches to its correctness proof.

REFERENCES

1. M. Bassiouni, "Single-site and Distributed Optimistic Protocols for Concurrency Control" IEEE Transactions on Software Engineering, vol. SE-14, no. 8, pp. 1071-1080, August 1988.
2. R. Bayer, M. Heller, and A. Reiser, "Parallelism and Recovery in Database Systems", ACM TODS, vol. 5, no. 2, June 1980.
3. D. Bell and J. Grimson, Distributed Database Systems, Addison-Wesley, 1992.
4. P. A. Bernstein, N. Goodman, and M. Y. Lai, "Analysing Concurrency Control when User and System Operations Differ", IEEE Transactions on Software Engineering, vol. SE-9, no. 3, pp. 233-239, May 1983.
5. P. A. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.
6. Y. Breitbart, H. Garcia-Molina, and A. Silberschatz, "Overvview of Multidatabase Transaction Management", The VLDB Journal, vol. 1, no. 2, October 1992.
7. G. Buckley and A. Silberschatz, "Obtaining Progressive Protocols for a Simple Multiversion Database Model", Proceedings of the International Conference on Very Large Data Bases, pp. 74-81, 1983.
8. G. Buckley and A. Silberschatz, "Beyond Two-Phase Locking", Journal of the ACM, vol. 32, no. 2, pp. 314-326, April 1985.
9. S. Ceri and G. Pelagatti, Distributed Databases: Principles and Systems, McGraw-Hill, 1984.
10. C. J. Date, and H. Darwen, A Guide to the SQL Standard, Addison-Wesley, 1993.
11. C. J. Date, An Introduction to Database Systems, Addison-Wesley, 1995.
12. P. Franaszek and John T. Robinson, "Limitations on Concurrency in Transaction Processing", ACM TODS, vol. 10, no. 1, March 1985.
13. H. Garcia-Molina and K. Salem, "Sagas", ACM SIGMOD International Conference on Management of Data, San Francisco, CA, May 1987.
14. J. Gray and A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993.

15. H. F. Korth and G. Speegle, "Formal Model of Correctness Without Serializability", Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 379-386, 1988.

16. H. F. Korth and G. Speegle, "Long Duration Transactions in Software Design Projects", Proceedings of the International Conference on Data Engineering", pp. 568-575, 1990.

17. H. F. Korth and A. Silberschatz, Database System Concepts, McGraw-Hill, 1991.

18. D. Kumar, "A Novel Approach To Sequential Simulation", *IEEE Software*, vol. 3, no. 5, pp. 25-33, September 1986.

19. D. Kumar, S. S. Iyengar, and M. B. Sharma, "Corrections to a distributed depth-first-search algorithm", *Information Processing Letters*, vol. 35, pp. 55-56, 15 June, 1990.

20. D. Kumar, "Systems With Low Distributed Simulation Overhead", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 155-165, March 1992.

21. D. Kumar, "Development of A Class of Distributed Termination Detection Algorithms", *IEEE Transactions on Knowledge And Data Engineering*, vol. 4, no. 2, pp. 145-155, April 1992.

22. D. Kumar, "A class of systems with nearly zero distributed simulation overhead", *Information Sciences, An International Journal*, Elsevier Science Publishing Co., New York, vol. 66, No. 1 and 2, pp. 23-42, December 1, 1992.

23. D. Kumar, "Efficient distributed simulation of acyclic systems", *Information Sciences, An International Journal*, Elsevier Science Publishing Co., New York, vol. 66, No. 1 and 2, pp. 167-190, December 1, 1992.

24. D. Kumar, and S. Harous, "A Study of Achievable Speedup in Distributed Simulation via NULL Messages", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 3, pp. 347-354, March 1993.

25. D. Kumar, and S. Harous, "Distributed Simulation of Timed Petri Nets: Basic Problems and Their Resolution", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 10, pp. 1498-1510, October 1994.

26. D. Kumar and A. Silberschatz, "A Counter Example To An Algorithm For The Generalized Input-Output Construct Of CSP", *Information Processing Letters*, vol. 61, no. 6, pp. 287, March 28, 1997.

27. D. Kumar, "A Simple, Multiversion Concurrency Control Protocol For Internet Databases", Final Report for the Summer Faculty Research Program, August 1977.

28. D. Kumar and S. S. Iyengar, "A Semiformal Correctness Proof of a Distributed Depth-First-Search Algorithm", *Journal of Parallel and Distributed Computing*, in press.

29. L. Lamport, "What Good Is Temporal Logic?", *IFIP*, Information Processing 83, R. E. A. Mason (ed.), Elsevier Science Publishers B. V. (North-Holland), 1983.

30. C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwartz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging", ACM

TODS, vol. 17, no. 1, March 1992.

31. P. E. O'Neil, "The Escrow Transactional Method", ACM TODS, vol. 11, no. 4, December 1986.
32. M. T. Ozsu and P. Valduriez, Principles of Distributed Database Systems, Prentice-Hall, 1991.
33. C. Papadimitriou, The Theory of Database Concurrency Control, Computer Science Press, 1986.
34. D. Reed, "Implementing Atomic Actions on Decentralized Data", ACM Transactions on Computer Systems, vol. 1, no. 1, pp. 3-23, Feb. 1983.
35. A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Database Systems, ACM Computing Surveys, vol. 22, no. 3, pp. 183-236, 1990.
36. E. Simon, Distributed Information Systems, McGraw Hill, 1996.

AN AUTOMATED 3-D SURFACE MODEL CREATION MODULE FOR LASER SCANNED POINT DATA

Joe G. Chow
Associate Professor
Department of Industrial and Systems Engineering

Florida International University
University Park Campus
Miami, FL 33199

Final Report for:
Summer Faculty Research Extension Program
Warner Robins Air Logistics Center

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

And

Robins Air Force Base

January 1999

AN AUTOMATED 3-D SURFACE MODEL CREATION MODULE FOR LASER SCANNED POINT DATA

Joe G. Chow
Associate Professor
Department of Industrial and Systems Engineering
Florida International University

Abstract

This project aimed at the development of a software system that can automatically create a 3-D model of a sample part from laser scan point data. Due to time and funding constraints, this research focused only on those parts that were made up of standard surfaces, such as planes, cylinders, spheres and cones. This work also assumed that the input point data, upon which the surface model was based on, was already complete, accurate, and evenly-distributed.

For standard surfaces, some geometric characteristics can be identified. For example, when a sphere is cut by a planar surface from any direction, the intersected geometry will be a circle (or an arc for a partial sphere). Using these geometric characteristics, the type of standard surface in the point cloud can be determined and the corresponding surface parameters can be calculated. In this research, the following aspects were thoroughly explored based on the idea mentioned above:

- obtaining the geometric characteristics from point data;
- calculating the surface parameters according to the geometric characteristics;
- growing the surface beginning from a seed point;
- establishing a complete geometric model

To reduce software development efforts, this research utilized the reverse engineering software, SURFACER, to pre-process the point data to obtain a complete point cloud model. The point model was subsequently outputted and processed by the in-house developed software for the automated creation of a CAD model. The geometric CAD model, once completed, was inputted into SURFACER again for displaying and checking.

A sample part was used to test the accuracy of the software system developed. The results indicate that the methodology proposed in this work is technically sound. Compared to current model conversion methods (such as edge-based and face-based), the proposed methodology has the advantage of directly extracting surface features without using an iterative procedure.

AN AUTOMATED 3-D SURFACE MODEL CREATION MODULE FOR LASER SCANNED POINT DATA

Joe G. Chow

1. Introduction

Conventional engineering transforms engineering concepts and models into real parts, while in reverse engineering real parts are transformed into engineering models and concepts. Reverse engineering typically starts with measuring an existing object so that a surface or solid model can be deduced.

A flow chart of a typical reverse engineering process is shown in Figure 1. This process begins with digitizing an engineering component using a measuring tool to obtain surface coordinate points. These data points, after cleaning and smoothing, are used to construct a surface and/or solid model. The reconstructed CAD model serves as an input for subsequent design and manufacturing functions, such as finite element analysis, process planning, NC code generation, and fixture design.

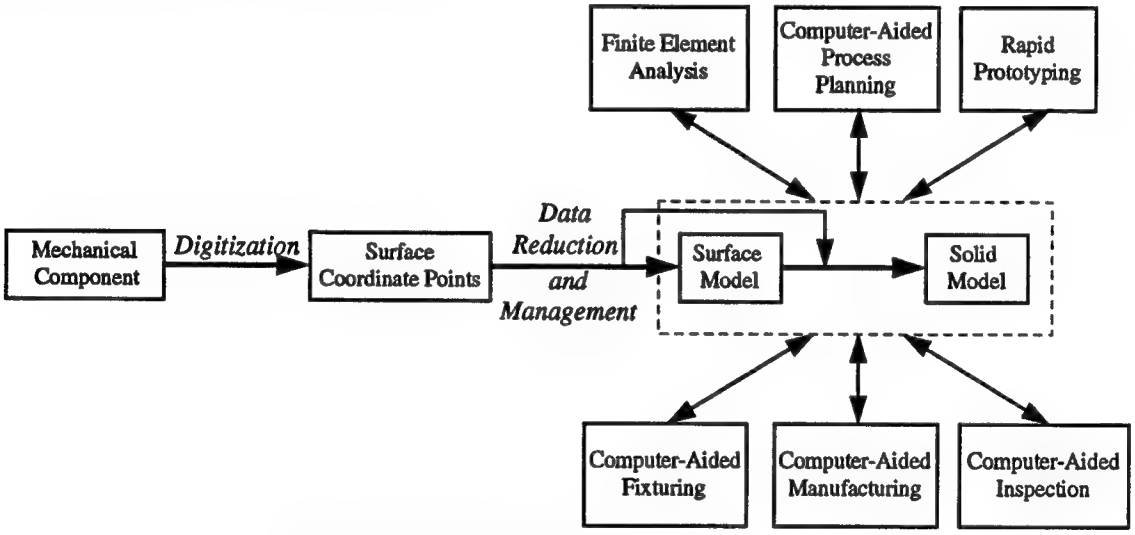


Figure 1. Reverse Engineering Methodology

Capturing shape and translating it into a CAD model is a difficult and complex problem. To obtain an accurate model by surface measurements through manual tools or a coordinate measuring machine is extremely time consuming. To reduce this effort, traditional methods usually sample only a few data points in a given area, which often results in less accurate and useful model. Recent advances in laser-based scanning and supporting technologies, capable of collecting a large number of data points in very short period of time, have provided a potential solution to reverse engineering.

Laser scanning is a non-contact procedure that provides fast and consistent acquisition of component geometry data through the use of a laser beam. In laser scanning, the Z-axis values are measured on a grid of predetermined X and Y coordinates. The depth information is provided by triangulation of the reflected light on a receptor mounted at an angle to the incident light. To acquire images from different views, the sensor may sweep over the part surface, or the part is moved under a fixed beam. At the end of scanning, a point cloud data is obtained for one or more scanned surfaces of the part. Figure 2. Shows the basic element of a laser system.

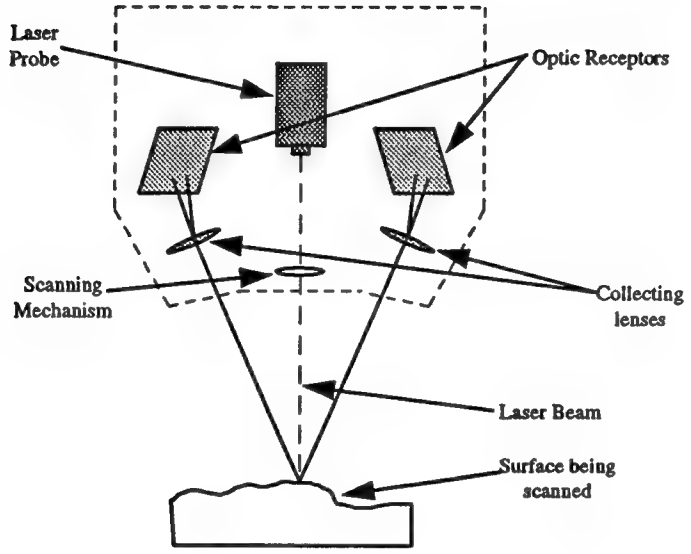


Figure 2. A Laser Scanning System

With the advent and increasing use of laser scanning techniques, there is an increased need for developing novel methods for geometric model reconstruction. In the case of reverse engineering, the point cloud data for all surfaces are edited and merged into a single point cloud data for the entire part through registration and alignment tools defined during scanning. A commercial reverse engineering software, such as Surfacer or Cimatron, is then used to convert the point cloud data into a surface or solid model by an experienced CAD operator. During model creation, the operator first connects boundary points into boundary curves and then transforms boundary curves into surface patches. When all surfaces are generated, a 3-D surface model for the entire part is obtained.

WR-ALC is one of the five Air Logistics centers of the United States Air Force (USAF). One of its primary responsibilities is to manufacture replacement and spare aircraft structural parts for the USAF. A majority of the machined structural components are characterized by volumetric features such as slots, deep pockets and cavities, and the demand for these components is typically in small quantities. Normally, WR-ALC aims to re-manufacture parts within tolerances (~ 0.010") without any design modifications or analysis.

Currently, manufacturing a part at WR-ALC involves creation of a 3-D CAD surface model, process planning, NC code generation, fixture design, and finally machining of the part. Though all the parts that need to be manufactured are accompanied by 2-D part drawings, and in most cases, a copy of the actual part, a CAD model of the part rarely exists. The 3-D CAD model needs to be manually created from the 2-D drawings or by direct measurement of the part. This process is usually the most time-consuming portion of the entire manufacturing process.

All the manufacturing activities at WR-ALC are currently performed manually. Due to the extent of human involvement, it requires on average six weeks to manufacture parts of medium complexity. In the case of high-complexity parts, the turnaround time may be several months long. To increase combat readiness and sustainability, it is essential for WR-ALC to reduce part turnaround time and costs. This requires automation of some of the steps in the manufacturing process.

Since CAD model creation constitutes a major portion of the turnaround time, maximum returns can be obtained by increasing the efficiency of this process. In the previous three summers ('95-'97), the principal investigator studied the feasibility of using laser scanning to reduce the model creation time for aircraft structural components. In those studies, sample parts, such as a leading-edge rib, forward latch fitting, and canopy fitting were scanned by laser scanners and surface models were reconstructed from the scan data. The accuracy of the reconstructed models were compared to original samples. The comparisons showed that the errors in the reconstructed model were on the order of 0.005", within the desired tolerances (0.010").

The feasibility study also demonstrated that the modeling time and skill level required to rebuild the model were significantly less than those required for the manual model creation. This was because the laser-based procedure began with a point cloud data which already captured all the dimensions and geometric features of the part. The feasibility study clearly indicated that the laser scanning was mature enough to capture and reproduce intricate surface details while satisfying the accuracy requirements of aircraft structural components. Consequently, WR-ALC is seriously considering implementing a laser-based reverse engineering system on their production floor.

In spite of significant reduction in part turnaround time, the modeling process using laser scan data is still very time consuming because of a great deal of human judgment and assistance. As a result, it usually takes days to convert point cloud data to a surface model. To further reduce the modeling time and enhance the benefits of laser scanning, this research aims to completely automate the model creation process.

The objective of this study is to develop a software module capable of converting the laser point cloud data into a 3-D surface model. In this work, the point cloud data are first inputted into the commercial reverse engineering software: Imageware's Surfacer. Interface programs are developed to manipulate the point data stored in the CAD

database to identify boundary curves and generate surface patches. A complete 3-D surface model is built when all the surface patches in the model are generated and connected.

The commercial Surfacer software has been chosen for this work because it is the most popular reverse engineering software on the market. It possesses excellent graphic capabilities and user interfaces. Interfacing with a commercial reverse engineering software like Surfacer will allow the principal investigator to concentrate his efforts on developing algorithms for curve and surface creation from laser point data, instead of on graphics and user interfaces.

2. Literature Review

The methods required to automatically reconstruct a surface model are reviewed in this section. Before raw point data is used to generate curves and surfaces, it is necessary to pre-process them with a specific reconstruction step. There are several practical methods which can pre-process point data, the major ones being: conversion (i.e., triangulation), reduction, noise elimination, alignment, and restoration. These processes are not necessarily performed in succession. Once preprocessed, the point cloud will be segmented and fitter with proper surfaces. In this section, methods of segmentation and surface fitting are discussed.

2.1 Segmentation

Segmentation has a direct bearing on geometric model reconstruction. Its objective is to logically divide the original point set into subsets, one for each natural surface, so that each subset contains just those points sampled from a particular surface, the process is called segmentation. Segmentation can be classified into two basic categories : edge-based and face-based .

Edge-based method attempts to find edge curves in the point data, and infers the surfaces from the implicit segmentation provided by the edge curves. Once sharp edges are being sought, we must try to find places where surface normals estimated from the point data change direction suddenly, while if smooth edges are met, we will need to look for places where surface curvatures or other higher derivatives have discontinuity. A representative example of this approach is described by Smith and Kanade [1]. Another edge-based segmentation technique is presented by Milroy et al [2].

Edge-based techniques suffer from the following problems. Sensor data, especially from laser-based scanners, are often unreliable near sharp edges, because of specular reflections there. The number of points used for segmenting the data is small, i.e. only points in the vicinity of the edges are used, which means that information from much of the data is not used to assist in reliable segmentation. In turn , this means a relatively high sensitivity to occasional

spurious data points. Because computation of derivatives from noisy point data is error prone, finding smooth edges is very unreliable.

Compared with edge-based approach, face-based techniques have several advantages. Face-based techniques try to infer connected regions of points with similar properties as belonging to the same surface (e.g. groups of points all having the same normal belong to the same plane), with edges then being derived by intersection or other computations from the surfaces. They work on a larger number of points, in principle using all available data. Deciding which points belong to which surface is a natural by-product of this method, whereas with edge-based method, it may not be entirely clear to which surface a given point belongs even after we have found a set of edges. Typically, the face-based approach also provides the best-fit surface to the points as a by-product.

2.2 Surface Fitting

Surface fitting can be divided into two classes: bottom-up and top-down methods. Bottom-up methods starts from seed points (selected as start point). Small initial neighborhoods of points around them, which are deemed to consistently belong to a single surface, are constructed. Local differential geometric or other techniques are then used to add further points which are classified as belonging to the same surface. When there are no more 'consistent' points in the vicinity of the current region, growing stops. Typically, several such regions may be grown in parallel, when they grow enough to touch each other, we find that they are compatible and can be merged into the same surface.

Practically, during the growing phase, we may also have to be prepared to update our idea of what surface the region represents. A set of points comprising a region may initially be well represented by a plane, but as more points added, it may be necessary to assume instead that the points in the region belong to a cylinder of large radius. The region is kept, but our idea of what underlying surface can change. Good examples of the state-of-the-art in region growing can be found in Leonardis et al [3] .

Top-down method goes in the opposite order, it starts with the premise that all the points belong to a single surface, and then tests this hypothesis for validity. If hypothesis is true, the process is done. Otherwise the points are subdivided into two (or more) new sets, and the single-surface hypothesis is applied recursively to these subset. The process continues until all generated subsets satisfy the hypothesis.

Various problems exist for both of these two approaches. In the bottom-up case, choosing good seed points to start growing the surface can be difficult. We need to decide whether to distribute the seed points uniformly, or sophisticated, we need to carefully choose which surface type to use for a region if more than one type of surface is

being considered and when to change the surface as the region grows. Deciding whether to add points to the region can be difficult, if bad points are wrongly added to the region, this will distort current estimates of the nature of that surface.

For top-down approach, one major problem is to choose where and how to subdivide. Often, the subdivision will be done along a straight line rather than a 'natural' boundary, and so in practice, merging steps are also required to re-combine pieces. This leads to edges which are rather jagged in nature. When surfaces slowly and smoothly blend into one another, the subdivisions chosen may be quite some way from the real boundary, and so a lot of extra work may be done subdividing in the wrong place.

Finally, both approaches, unless carefully controlled, are likely to end up representing a complex free-form surface of a type that is not included in this study as many small pieces of planar or quadric surfaces, which is not the desired result.

3. Methodology

To create a geometric model based on point cloud data, the point clouds first need to be segmented into a series of patches that correspond to the surfaces of the duplicated part. These patches will then be fitted with the appropriate curves and surfaces so that a complete CAD model can finally be created. In this research, neither method will be strictly employed for point cloud segmentation. Instead, certain *geometric characteristics* will be directly extracted from the point cloud data, from which the surface type can be determined. The point cloud can then be segmented according to these identified surface types.

3.1 Geometric Characteristics and Surface Parameters

For each of the standard features considered in this research (plane, cylinder, sphere, torus, and cone), there exist certain defining geometric characteristics. For example, the cylinder has a central axis, whose planar cross section (intersection between a plane containing the central axis and the cylinder itself) is a straight line parallel to the axial direction. Its perpendicular planar cross section (intersection between a plane perpendicular to the central axis and the cylinder itself) is an arc or a circle, depending on whether the feature has the shape of a portion of cylinder or of a complete one, respectively.

The combination of these two geometric characteristics (a straight line and an arc/circle, intersecting and perpendicular to one another) would thus identify a cylinder. In the same way, related geometric characteristics

based on these two perpendicular planar cross-sections can be defined to identify all standard surface types. These shape-specific characteristics are shown in Figure 3 for each of the standard features investigated in this research.

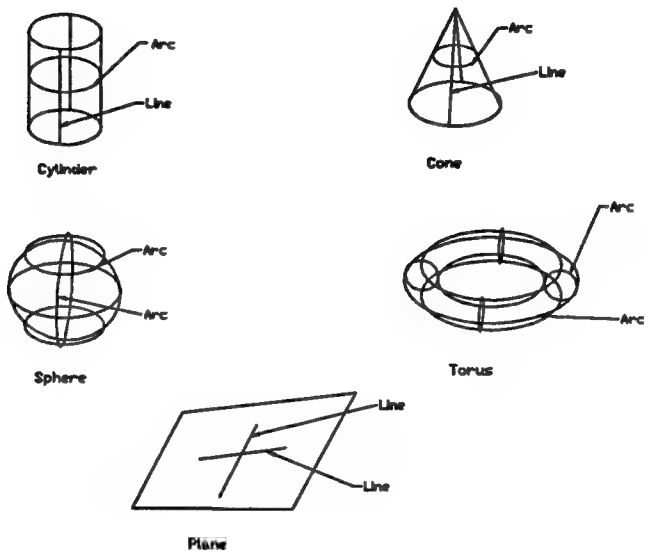
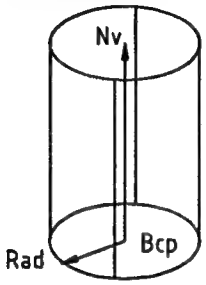
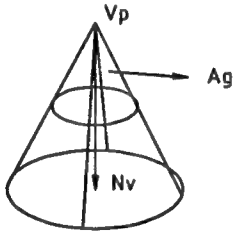
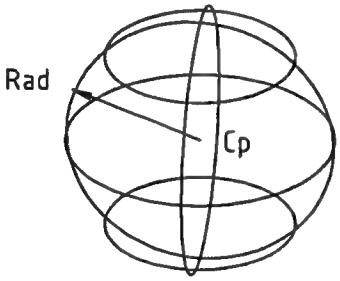
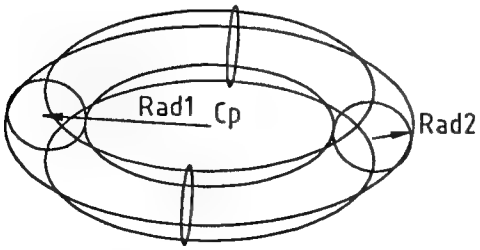


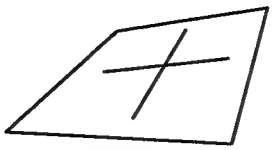
Figure 3. Identifying Geometric Characteristics of Standard Surfaces

Thus, if points can be extracted that constitute one of the sets of geometric characteristics shown above, the corresponding standard surface type can be determined. Based on the geometric characteristics, the parameters for the identified standard surface can then be calculated. The parameters for each of the standard surfaces considered in this research are listed in Table 1.

Freeform surfaces can conceivably contain the same points that would form the geometric characteristics of standard surfaces. Therefore, to ensure that the point cloud data represents the predicted standard surface, other points in the point cloud data apart from the geometric characteristic points should be analyzed. Using the parameters determined above, these points can be tested for whether they lie on the predicted standard surface. If enough groups of points are tested and found to lie on the standard surface determined by the identified geometric characteristics, then the determination can be concluded to be correct and the appropriate surface can be segmented. The threshold value for the number of points to be tested (above which the surface can be concluded with confidence) can be set in proportion to the size of the predicted surface.

Table 1: Parameters for Standard Surfaces

Surface type	*Parameters*
 Cylinder	Center of the Base, B_{Cp}; Normal Vector of the Central Axis, N_v; Radius, Rad
 Cone	Vertex, V_p; Angle of Cone, *Ag*; Normal Vector of the Central Axis, N_V.
 Sphere	Center, C_p; Radius, Rad.
 Tour	Center, Cp; Major Radius, Rad_1; Minor Radius, Rad_2.

 Plane	$A \cdot x + B \cdot y + C \cdot z + D = 0$

Therefore, if the relevant geometric characteristics can be extracted from the point cloud data, the type of surface to which the points belong can be determined. By calculating the corresponding parameters and using them to test other points, that area of the point cloud can be identified as a specific standard surface with known parameters. If testing fails for the other points, the surface can at least be identified as not having standard parameters, and is thus a freeform surface.

If the parameters are proved correct and a standard surface is identified, those points that lie on this surface are processed and grouped. The geometric characteristics of other surfaces in the point cloud are then extracted, from which the corresponding surface parameters will also be calculated and tested. In this way, all points can be grouped together according to the surface on which it lies, and the appropriate topological information can be extracted. The entire point cloud can thus be segmented accordingly.

Since segmentation is the bulk of the burden in developing an automatic creation process for geometric models, the remaining tasks (such as curve fitting and surface fitting) can be easily accomplished once the surfaces and their boundaries are found.

3.2 Architecture and Procedure

The entire procedure for digitization-based geometric model creation is shown in Figure 4. Generally, point cloud data obtained from digitization (such as those realized using a laser scanner or mechanical touch probe) are divided into groups of files according to each of the various setups used. In addition, the point cloud data is often not uniform and the size too large.

Thus, the point clouds must first be pre-processed to produce a single, complete, consistent and concise point cloud model, such that subsequent processing can be properly performed. The pre-processed points are segmented into a series of patches according to related surfaces. Based on these patches, the surfaces and boundary curves of the point cloud can then be fitted, producing the desired geometric model. This research will only focus on surface model and B-rep model creation.

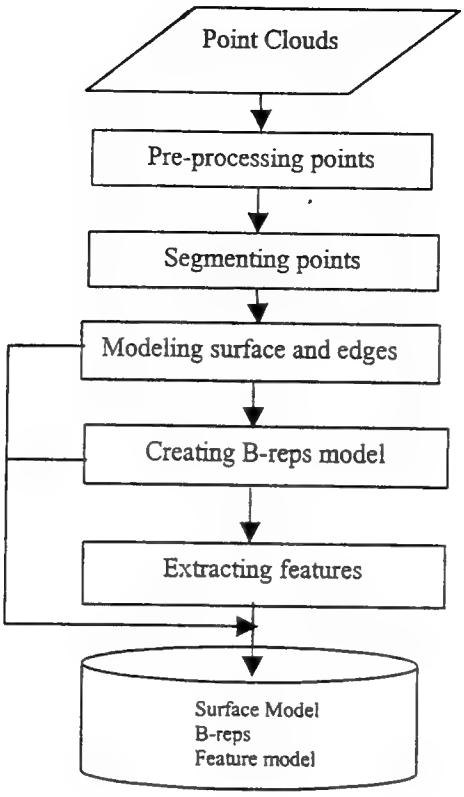


Figure 4. Procedure for Geometric Model Creation

The system architecture for the implementation of the entire procedure for the geometric model creation is shown in Figure 5. Pre-processing operations, such as sampling and registration of the point clouds, will be performed interactively within SURFACER. When a satisfactory point cloud model is obtained, the data will be outputted into a relationship database using a Hash table mode, in which the points are grouped into cells whose position is represented by ordered triples *(i, j, k)*. This method of organization would expedite subsequent procedures by allowing for a convenient and standardized procedure for accessing points. Based on this database, both point cloud segmentation and geometric model creation can be smoothly executed, under the VB/VC and SQL server. Results will be displayed in SURFACER.

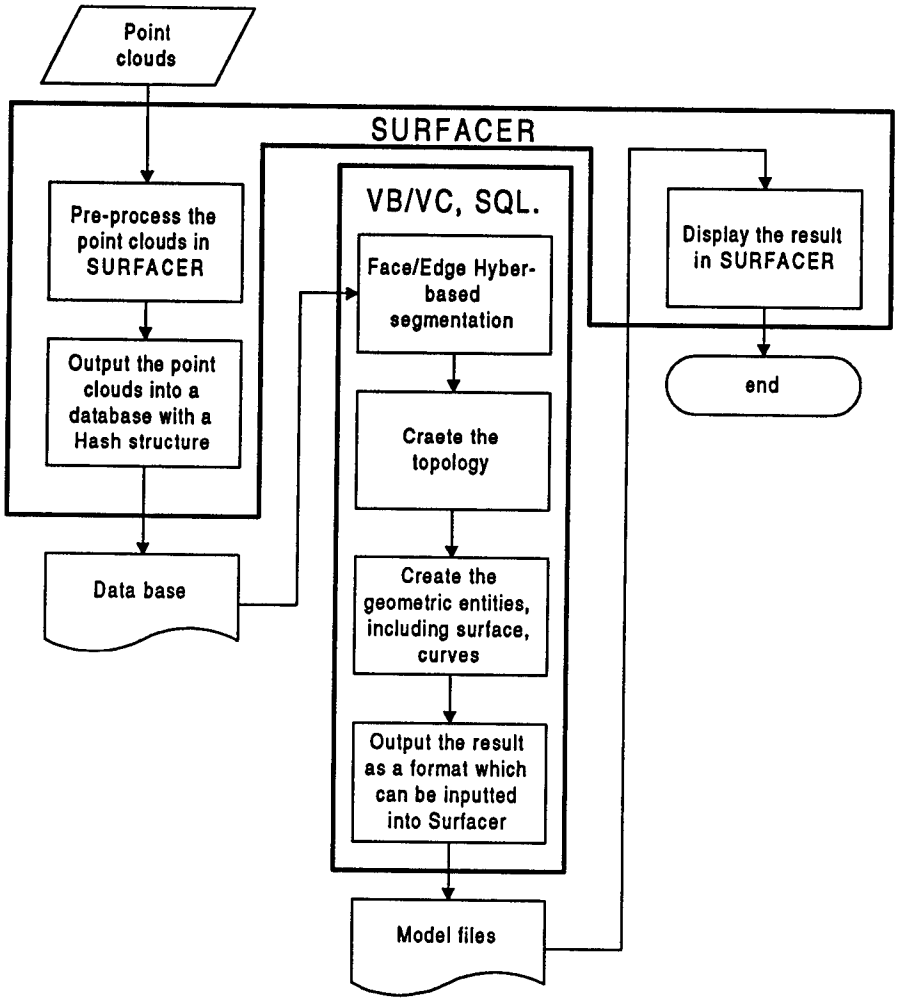


Figure 5. System Architecture

3.3 Information Model and Data Flow Chart

There are two information models, one for point cloud segmentation and one for B-rep model creation. For the segmentation information model, the pre-processed points will be stored as a *cell information structure* and a *point list*. The segmentation results will then be recorded in a *segmented face list*. The topological information detected after the completion of segmentation will be stored in a temporary structure labeled as *intermediate information data*. Based on this information, the data for the B-rep information model can created, represented as a *face list*, *loop list*, and *edge list*.

3.4 Description of Individual Algorithms

The algorithms related to each module listed in Figure 5 will be described in detail in this section according to their relative order of operation. Because pre-processing of the point clouds is straightforward and is performed using SURFACER's existing functions, it is not described here. Thus, a complete and consistent point cloud model is already assumed for use in the following algorithms.

3.4.1 Outputting the Point Cloud as a Hash Atructure into a RDB file

The point cloud data is to be first organized as a Hash structure under SURFACER to allow for convenient access of individual point data, thus increasing efficiency in future operations. However, SURFACER's SCOLL function is not able to directly access a relationship database (RDB) file. The point cloud data must be outputted into a single file under SURFACER, and a VB program is then used to convert this file into a RDB file.

To create the Hash structure, a rectangular box is first created around the point clouds and is divided into cubic unit cell meshes. The data points are contained within these individual cubic units and are thus organized in a *cell-occupied matrix*. Each cell is assigned an ordered triple (i, j, k) with values according to the cell's position in each of three dimensions. Such a standard method of organization would allow particular cells (and adjacent cells, if desired) to be easily called up according to their index numbers.

The procedure for creating this Hash-structure RDB file is described in greater detail in the following subsections.

3.4.1.1 Organizing the Point Cloud as a Hash Structure

SURFACER provides a SCOLL function that allows the user to obtain the *bounding box* of a point cloud. Using this function, the coordinates of the two critical bounding points are extracted: the lowermost left and uppermost right points of the point cloud. An algorithm is written to create the desired cell-occupied matrix based on these two points as the two opposite corners of the bounding box. This algorithm is implemented in SCOLL, and its flowchart is shown in the Appendix. A description of its utilization follows.

The coordinates of the two corner points are given as (x_1, y_1, z_1) and (x_2, y_2, z_2) for the left-lower and right-upper points respectively. Based on the part size, point cloud accuracy, and model precision requirement, the cell size (C_{size}) is specified interactively by the user. The origin of the matrix coordinate system is then defined at a point offset outwards from the left-lower corner coordinate, according to the following equations:

$$x_0 = x_1 - C_{size} / 2$$

$$y_0 = y_1 - C_{size} / 2$$

$$z_0 = z_1 - C_{size} / 2$$

Offsetting the point of origin for the matrix index to (x_0, y_0, z_0) in this way ensures that all points in the point cloud will be included within the matrix, and thus all points will lie within one of its individual cells. With this point of origin as its initial corner coordinate, a rectangular structure made up of a series of individual cubic unit cells of a particular C_{size} can be made, thus producing the cell-occupied matrix. Each point, with coordinates (x, y, z), in the point cloud is then selected in turn to determine in what cell it lies, according to the following equations:

$$i = \text{int}[(x - x_0) / C_{size}]$$

$$j = \text{int}[(y - y_0) / C_{size}]$$

$$k = \text{int}[(z - z_0) / C_{size}]$$

A point (x, y, z) would occupy the cell indexed by (i, j, k), where i, j, and k are the integers obtained from the equations above. (The left-lower corner point would thus lie in the cell of origin, indexed as (0,0,0).)

The resulting cell-occupied matrix is a sparse matrix, as many of its cells do not contain any points and are empty. By expressing the matrix as a Hash structure, the amount of memory required to represent it can be dramatically reduced. Since the index (i, j, k) is unique for each cell in the matrix, the string "i·j·k" can be used as the entrance key to the Hash table. When more than one point occupy the same cell, they are linked together under one index key.

3.4.1.2 Outputting the Hash Table Data into a File under SURFACER

Once organization of the cell-occupied matrix has been completed, it is outputted into a file using the following format:

KEY_1, N_1

$p_{11}, p_{12}, \cdots, p_{1N1}$

...

KEY_m, N_m

$p_{m1}, p_{m2}, \cdots, p_{m\,Nm}$

where KEY_a is the corresponding key string i·j·k for any cell *a*, N_a is the number of points in the cell indexed by KEY_a, and p_{ab} are the coordinates (x, y, z) of a specific point *b* within cell *a*.

3.4.2 Segmenting the Point Cloud

Point cloud segmentation is the most significant step in automatic geometric model reconstruction. In this module, the function of the each of the primary algorithms are: 1) to locate a *seed point* that can be used to extract the

geometric characteristics of the surface; 2) to obtain the geometric characteristic points along the planar cross-section in each orthogonal direction; 3) to calculate the parameters of the surface; and 4) to group all points which belong to this surface.

Once the first surface is identified and its points are grouped for segmentation, its boundaries are used to select another seed point upon which geometric characteristic extraction for another surface will be based. This procedure is repeated until all points in the point cloud have been processed and grouped, and thus all surfaces have been segmented out.

3.4.2.1 Selecting a Seed Point

As a general rule, the seed point should not lie on or near a surface boundary in order to avoid geometric ambiguities, as shown in Figure 5. The normal vector at the seed point needs to be calculated later for geometric characteristic extraction, and the appropriate normal vector may not be available from calculations based on boundary points. Thus, a selected point should be tested to determine whether it lies too close to a boundary, and whether a new point should be selected as the desired seed point.

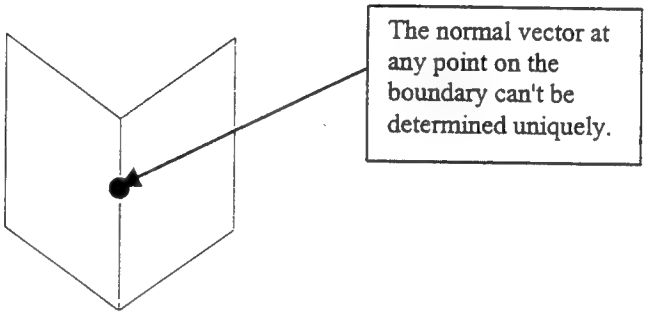


Figure 5. The ambiguities of normal vector at a point

There are two situations to be discussed when considering seed point selection. The first is when a seed point is being selected for the very first surface, at the beginning of the segmentation procedure. The second situation deals with the subsequent selection of seed points for other surfaces in the segmentation cycle, based on the boundaries of previously defined surfaces.

In the first situation, a point is chosen (preferably with the highest k- index number of the cell matrix) to be used as the initial seed point of the procedure. To test whether the point lies on (or too close to) the boundary of a surface, the change in its curvature in relation to adjacent points in each of four orthogonal directions is analyzed.

(Adjacent points are considered to be those points adjacent in the direction of the x-axis or y-axis, thus having consecutive values for their i- and j- index numbers, respectively.) A dramatic change in curvature along any direction would indicate a nearby boundary, and thus another point in the direction opposite to that of the discontinuous curvature should be chosen. If the curvature in all directions is continuous, the selected point is acceptable to be used as the seed point for geometric characteristic extraction.

Once segmentation is completed for the surface on which the initial seed point lies, the second situation is encountered, when a second seed point is to be determined based on the series of boundary points obtained for the first surface. One of these boundary points is chosen, making sure that it is not a corner point. A neighborhood of a certain distance around that point is set, so that some of the points within that neighborhood lie on the same surface on which the initial seed point lies. Of the remainder of the points within the neighborhood that do not lie on this surface, a new seed point is selected. To ensure that the new seed point is not too close to the boundary, the following two parameters are observed, as shown in Figure 6.

1) The angle between the boundary line (within only a small neighborhood around the chosen boundary point) and the line joining the new seed point and the chosen boundary point should be as close to 90° as possible, i.e. at a maximum.
2) The distance between the new seed point and the chosen boundary point should be at a maximum.

Once segmentation is completed for this new surface as well, succeeding seed points for each of the other surfaces are then also selected in the same way as described here. A point on the boundary of a previously determined surface will be used to select a new seed point for a yet-undetermined surface.

3.4.2.2 Searching for the Geometric Characteristics

Once a seed point for a particular surface is selected, a *search path* is determined to seek the appropriate geometric characteristic. A plane is rotated around a *search path axis*, and its cross section with the surface is calculated, as shown in Figure 7. The curvature of the series of cross sectional points is analyzed to determine whether they comprise one of the desired geometric characteristics.

The search path axis is determined based on the normal vector at the seed point. The flowchart for the algorithm used to calculate the normal vector is included in the Appendix. First, several points adjacent to and surrounding the seed point are selected. Any two successive points among these would form a triangular face with the central seed point as the third vertex. With several triangular faces thus formed, each with the seed point and two of its

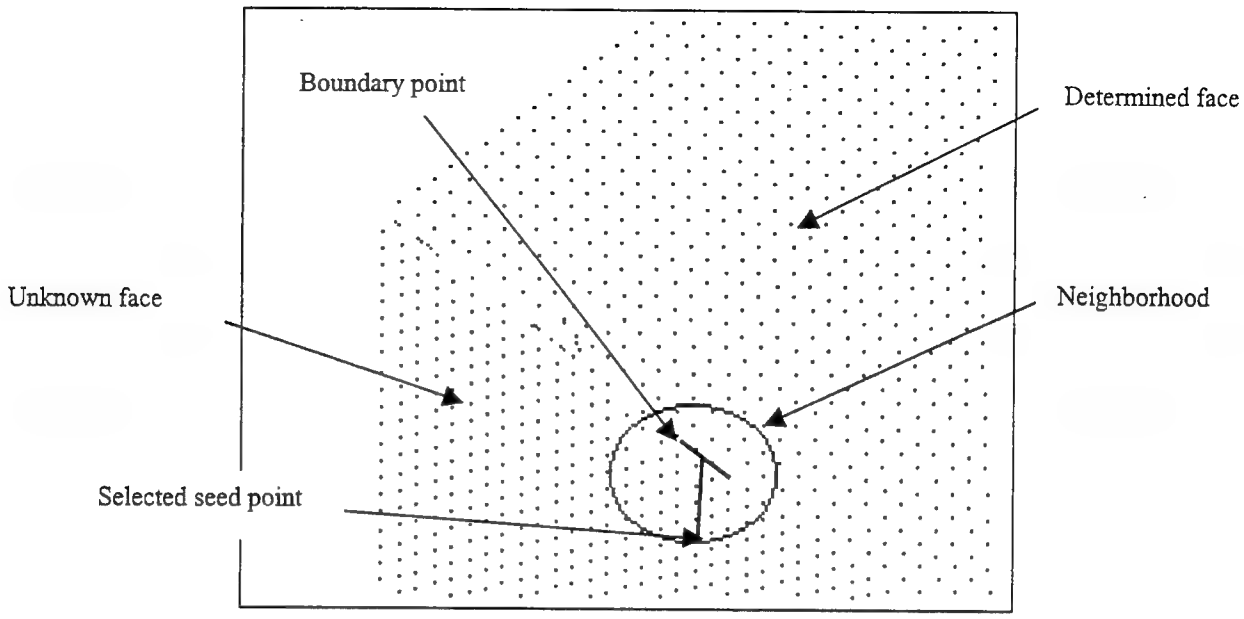


Figure 6 New seed point for defining unknown adjacent surface

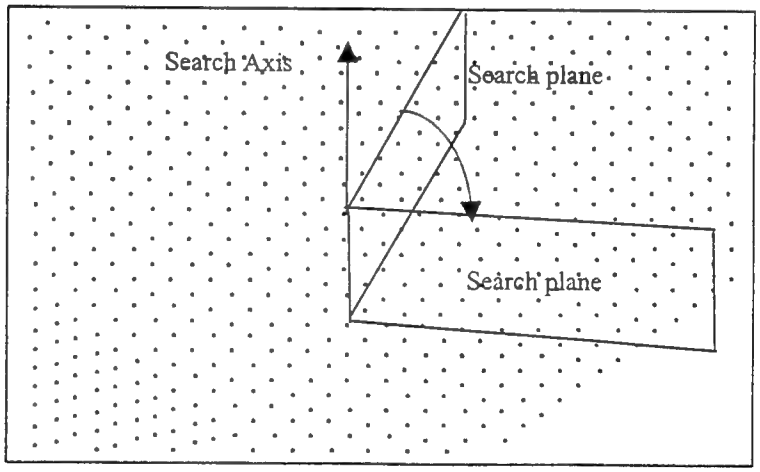


Figure 7. The search axis and search plane

adjacent points as its three vertices, the normal vector of each face is calculated by the equation:

for a triangle with vertices p_1, p_2, p_3:

$$N = \text{norm}[(p_2 - p_1) \times (p_3 - p_1))]$$

where norm signifies standardization to a unit vector.

In calculating each normal vector, it is noted that they should all face in the same general direction, i.e. the angle between any two vectors should not exceed 90°. The average of the vectors obtained from the various triangular faces would then give the surface normal vector at the seed point, given by:

for *n* adjacent triangles, the average normal vector N':

$$N' = 1/n \sum N_i \; (i = 1, 2, \ldots, n)$$

From the normal vector at the seed point, the search path axis is defined as the line containing and in the direction of this vector. A point adjacent to the seed point is chosen as the initial *search point*, and the plane containing both this search point and the search path axis is defined as the search path S(P) such that it satisfies the following matrix equation:

$$S(P) = \begin{vmatrix} x & y & z & 1 \\ x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ l & m & n & 0 \end{vmatrix} = 0$$

where (x_0, y_0, z_0) are the coordinates of the seed point, (l, m, n) is the normal vector, and (x_1, y_1, z_1) are the coordinates of the chosen search point, as shown in Figure 8.

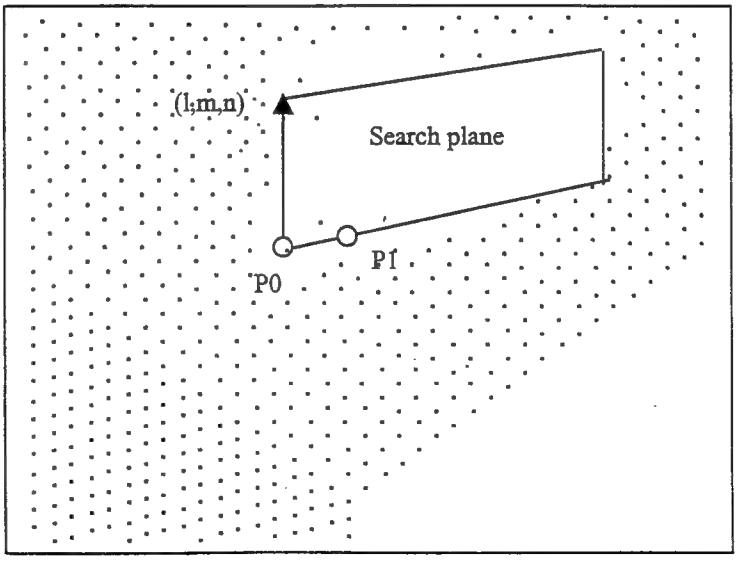


Figure 8. The parameters for defining the search plane

The intersection between this plane and the point cloud is determined by finding each point of cross section in sequence, one at a time. From the index value (a, b, c) of the search point, a neighborhood of adjacent points is first established as the collection

$$P = \{ \text{cell}(i, j, k) \mid (i = [a+1, a-1];\ j = [b+1, b-1];\ k = [c+1, c-1]) \}.$$

The points that lie on the planar cross section are determined as the points in this neighborhood that are closest to the defined search path, i.e. that satisfy the equation of the plane within a certain tolerance. The points are thus extracted by

$$P_{sec} = \{ P' \mid S(P') \leq \delta;\ P' \in P \}$$

where the value δ is experimentally determined.

A new neighborhood is then established based on one of these extracted points as the new search point, and other points along the search path are extracted in the same way along the same direction. This is repeated until all the points that sequentially comprise the desired planar cross section have been obtained.

To examine whether this extracted curve is a standard geometric characteristic, the curvature at each of its points is calculated according to the following equation:

for a point p, with the previous point p_p and the following point p_n:

$$C = \text{ang}[(p - p_p), (p - p_n)] / \text{dist}(p_p, p_n)$$

where ang signifies the angle between two lines, and dist means the distance between two points.

If all the points in the curve have equal curvature, then the curve is an arc or circle. If the curvature at each point is zero, then the curve is a line. Either of these two cases would signify that the extracted curve is one of the desired geometric characteristics, and the second geometric characteristic can be found by simply extracting those points along a search path perpendicular to the one previously used (its cross section with a perpendicular plane also passing through the search path axis).

If the curvature of each point is found to be relatively inconsistent, a new search path is determined by rotating the plane around the search path axis by a certain degree. The cross sectional points along this new search path are extracted and their curvatures are tested for consistency using the method described above. This is repeated until the curve is found to be either an arc/circle or a line. If neither of these is found within any of the trials through a complete rotation of 180°, then no geometric characteristic probably exists, and the surface is freeform.

3.4.2.3 Calculating the Surface Parameters

From the two geometric characteristic curves that are extracted, the standard surface type can be determined and its parameters (listed in Table 1 for each surface type) can be calculated.

If the geometric characteristics extracted are both lines, the surface is a simple plane. The plane's parameters (A, B, C, D) can then be easily calculated using any three points (that are not co-linear) from the two lines, as shown in Figure 9.

If the two curves extracted are a line and an arc, the surface is likely to be a cylinder. As its defining parameters, the cylinder's central axis and radius must be found. The central axis is the line derived from the arc's center coupled with the line's directional vector. The radius of the cylinder is assigned as the radius of the arc itself, as shown in Figure 10.

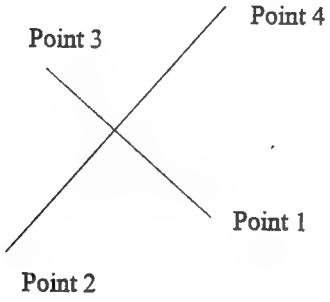


Figure 9. Calculating a plane's parameters

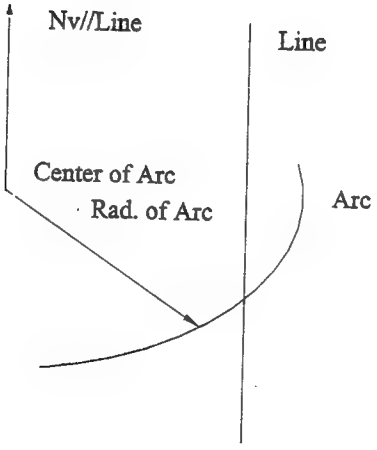


Figure 10. Calculating a cylinder's parameters

If both geometric characteristics are arcs, then the surface is likely to be either a sphere or a torus. For each arc, the line that passes through its center and is perpendicular to the arc's plane is found. If these two lines intersect, then the surface is a sphere. If they do not intersect and are skew (the distance between them is greater than some value Δ), then the surface is a torus, as shown in Figures 11 and 12 . For a sphere, the point of intersection of these two lines would give the center of the sphere. The sphere's radius can then be easily calculated as the distance between the center and any point on either of the two arcs. For a torus, the center is derived by first calculating the distance between the center of the minor arc (arc of smaller radius) and the plane of the major arc (arc of larger radius). Using this value, the center of the major arc is shifted the same distance along the direction of the major arc's normal vector, making sure to shift towards the same direction as where the center of the minor axis lies. This new shifted point is the center of the torus. (Thus, the center of the torus and the center of the minor arc should lie on the same side of the major arc's plane.) The major radius is the distance between the center of the torus and the minor arc's center, and the minor radius is the radius of the minor arc itself.

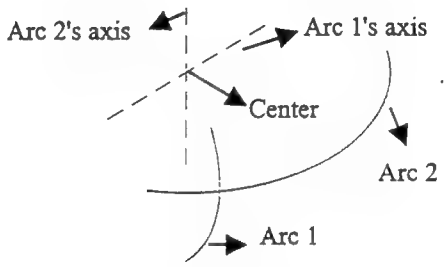


Figure 11. Calculating a sphere's parameters

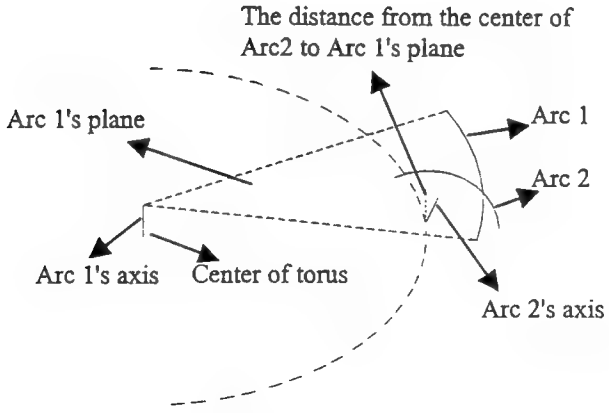


Figure 12. Calculating a torus' parameters

If one of the extracted characteristics is a line and one is a non-circular curve (with an apparently elliptical curvature), then the surface may be a cone. To determine the cone's parameters, a third curve should be extracted: an arc or circle (i.e., its cross section with a plane perpendicular to the cone's central axis). To find this third characteristic for the cone, the plane whose search path gave the arc characteristic is rotated by a certain degree and the new cross section is found. The points are then extracted and their curvature is tested as in the method described in Section 3.4.2 to determine whether the curve is an arc. If not, the plane is further rotated and the cross section is tested. Each plane, being rotated in increments, should still pass through the original seed point and should remain perpendicular to the plane used to determine the first search path (i.e., the one used to extract the line characteristic). If no such arc is found after searching by rotation in both directions, the surface can be considered to be freeform. If such an arc is indeed found, its center is calculated, along with the normal vector of the arc's plane. The central axis of the cone would then be the line passing through the arc's center and in the direction of its normal vector. The vertex of the cone is obtained by the intersection of this central axis and the line geometric characteristic, as shown in Figure 13.

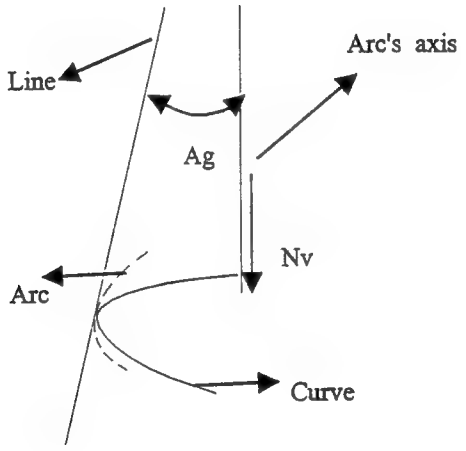


Figure 13. Calculating cone parameters

3.4.2.4 Grouping All Points Lying on The Surface

Once the parameters for the surface have been calculated, all the points that lie on this surface can be grouped together. The region of points lying on the surface can be grown by using the seed point as the origin. Starting from the seed point, all adjacent points are calculated for whether they lie on the surface described by the defined parameters. Successive points are then continuously searched by testing other points adjacent to previously searched points, growing outward from the original seed point. This is continued in each direction until a point is

encountered that is found to not lie on the predicted surface. This would indicate the beginning of a new surface, and thus the boundaries of the initial surface are formed. In this way, all points that have been searched and found to lie on the surface can be grouped together and segmented from the point cloud.

To determine whether a point lies on the predicted surface:

Supposing the surface's equation is

$$F(P) = 0$$

A point p would lie on the surface if it satisfies the inequality

$$F(p) \leq \delta$$

F(p) can be defined as an equation for the distance between a point p and the surface, so that a point lies on the surface if this distance is less than some tolerance value δ. According to the surface parameters of each standard surface type, this distance can be calculated as follows:

For a plane: The coefficients of the plane's equation (A, B, C, D) can be used to simply find this distance. The point is on the plane if it satisfies

$$\left| \frac{(A, B, C, D)\,(p_x, p_y, p_z, 1)^T}{(A^2 + B^2 + C^2)^{1/2}} \right| \leq \delta .$$

For a cylinder: The distance can be calculated using the distance between point p and the central axis A, dist(p, A), i.e. the distance between a point and a line. The difference between this distance and the radius R would give the distance between point p and the cylinder itself, according to

$$F(p) = | \text{dist}(p, A) - R | \leq \delta$$

For a sphere: The distance between point p and the center C is used. The difference between this value and the radius R is the distance between point p and the sphere, so that

$$F(p) = | \text{dist}(p, C) - R | \leq \delta$$

For a torus: The angle between the line joining point p with the center and the central axis (normal vector of the torus' plane at its center) is calculated and assigned as angle α. The distance between point p and the center is calculated as the value a. Along with the length of the major radius R, the distance between point p and the center of the torus' rim (center of minor arc) is then given by

$$\text{dist}(p, \text{center}) = [a^2 + R^2 - 2aR \cdot \cos(90 - \alpha)]^{1/2}$$

so that the distance between point p and the torus itself is the difference between this value and the minor radius r, such that

$$F(p) = | [a^2 + R^2 - 2aR \cdot \cos(90 - \alpha)]^{1/2} - r | \leq \delta$$

For a cone: The angle between the line joining point p with the vertex V and the central axis A is calculated. If this angle is within tolerance to the angle parameter θ of the cone, then point p lies on the surface, so

$$F(p) = | \text{ang}(p\text{-}V, A) - \theta | \leq \delta$$

Although the geometric characteristics were extracted, the point cloud is not necessarily the standard surface described by the calculated parameters. A freeform surface could also contain planar cross sections that fit the requirements of the desired geometric characteristics, even though the surface may be far from what the resulting parameters would indicate. Thus, when growing the region of points using the process described above, only a few points might be found to satisfy the parameters before a boundary is encountered. If the number of points found to lie on the surface is not beyond a certain threshold, then the surface can be concluded to be freeform. If the region is grown beyond the threshold, the surface is considered large enough to be segmented as a standard surface defined by the calculated parameters. The threshold of the number of points that the grown region should contain before the surface can be concluded with confidence is set according to the approximate size of the surface.

3.4.3 Detecting Topological Information

Once all the surface types of the point cloud have been determined and segmented into separate patches, its topological information needs to be detected. Adjacent surfaces should share only one series of points as their boundary. However, because of limited precision of the points themselves and from calculation error, there are likely to be gaps and overlaps between adjacent surface patches. A special algorithm is thus created to detect and define the boundary points of the point cloud in extracting topological information.

The gap and overlap between surface patches is assumed to not be beyond a particular neighborhood. Shared points between two adjacent patches are found using a *detection ball* with a radius equal to the size of the neighborhood. An initial point is chosen along the preliminary boundary of a particular surface (i.e. one of its outermost points), and the detection ball is centered on that point. Any points from the boundary of another surface that lie within the detection ball would indicate adjacency between the two surfaces. Other boundary points from the initial surface are then chosen in sequence along one direction to detect the extent of the adjacency, until a vertex is extracted (when points on the boundary of a third surface begin to lie within the detection ball). As each point along the boundary is processed, a shared point for the two adjacent surfaces is defined, taken as the average of both surfaces' boundary points. This procedure is repeated in the opposite direction from the initial chosen point as well, so that all shared points can be extracted along the entire boundary.

The process for boundary extraction described above is performed to extract the shared points for all the boundary edges of the point cloud, between all surfaces. Results are marked so that processing of boundary points is not needlessly repeated.

3.4.4 Constructing Surfaces and Curves

Once point cloud segmentation and boundary detection have both been completed, the geometric model can be created automatically based on the obtained information, reorganizing it using a B-rep format. The following information has been thus far determined in previous procedures: 1) all surfaces of the part; 2) each surface's boundary points; 3) adjacency of surfaces and their shared points.

3.4.4.1 Curves

The boundaries of each surface can be easily fitted with the appropriate curve by using the shared edge information obtained from boundary detection. The edge's vertices are known, along with what points lie on the edge and the type and parameters of the two faces sharing that edge. The curve can be constructed according to the following situations, using the two vertices of the edge as its two endpoints in each case:

If both faces are planes: The edge should be constructed as a simple line between the two endpoints.

If one face is a plane and the other a sphere: The edge should be constructed as an arc.

If both faces are spheres: The edge should be constructed as an arc.

If one face is a plane and the other a torus: If the plane is determined to be perpendicular to the central axis of the torus (the normal vector of the torus' plane at its center), then an arc should be used to fit the edge. Otherwise, a B-spline is used.

If one face is a plane and the other a cylinder or cone: If the plane is perpendicular to the central axis of the cylinder/cone, an arc is used to fit the edge. Otherwise, a B-spline should be used.

If one face is a sphere and the other a cylinder or cone: If the central axis of the cylinder/cone passes through the center of the sphere, an arc is used. If not, a B-spline is used.

If one face is a torus and the other a cylinder or cone: If their central axes coincide, the edge should be fitted with an arc. Otherwise, a B-spline should be used.

If one face is a cylinder and the other a cone: If their central axes coincide, the edge is fitted with an arc. Otherwise, it is fitted with a B-spline.

For all other situations: A B-spline is used.

3.4.4.2 Surfaces

The surfaces have already been created during segmentation, and so only the loops of the surface need to be constructed. Since the adjacency and edges between surfaces are already known, the information need only to be reorganized using a B-rep format. The flowchart for the algorithm employed to accomplish this is given in the Appendix.

3.4.5 Outputting the Model as a SURFACER-Compatible File

The results are difficult to display in VB/VC without supporting software. An alternative method is chosen, to display the results in SURFACER. The model is thus outputted into a special model file that be loaded in SURFACER using SCOLL, and can thus be displayed with SURFACER's existing entities. The key technique involved in the creation of this algorithm is the specification of the file format. It is defined below:

[Point START]
X_1, Y_1, Z_1
X_2, Y_2, Z_2
...
...
[Point END]
[Edge START]
Edge type, Edge point's list ($p_1, p_2,...$)
...
...
[Edge END]

[Face START]

Face Type, Face parameters, Face loop number *n*
$loop_1, loop_2, ..., loop_n$
[Face END]

3.4.6 Reading and Displaying the Model File using SURFACER

Since the model file already includes all the information representing the part's surfaces and boundary curves, it can be used as the basis for displaying the part in SURFACER using SURFACER's entities, such as surfaces and curves. SURFACER already provides numerous functions to represent surfaces and curves, along with various ways to display them. Therefore, this algorithm needs only to read the file and interpret it line by line, calling the related functions of SURFACER to display the corresponding surfaces and curves that comprise the part.

4. Implementation of the Algorithms

This section briefly discusses the implementation of the methodology outlined in the previous section. The primary focuses are : the development environment, the system architecture and information flow, and database structure and interface.

4.1 Development Environment

Generally, the points digitized by laser scan include some defects, and completely describing a part need a group of point cloud files obtained by scan from different direction. Therefore, the points data should be first pre-processed, such as register, filter, sample, and so on, in order to obtain a complete and perfect point cloud model. Additionally, the point number of a complete point cloud model is very huge, so that a database is needed to store and record the points and the processing result.

For the pre-processing of the point clouds, the reverse engineering software SURFACER is chosen as one of supporting software. SURFACER allows the user interactively process point clouds and establishes wire-frame or surface model. In this work, it was primarily used to prepare point data for automatic geometric model creation. The point data would be registered, filtered, and sampled by the user or programs, and then be outputted as a special format that can be stored into a database easily. After the geometric model is created automatically, the result model then can be inputted into SURFACER to display.

To realize the algorithms proposed in Section 3, and manage the point data via database, Visual Basic is used as the programming language because it provides the interface to call the Open Database Connection (ODBC) API. The ODBC also possesses the capability of calculating functions, creating database , and processing data items in the database.

Figure 14 shows the relationship between these supporting software. In the integrated environment, the point data digitized by laser scan is first processed in SURFACER by a user, and is then outputted into a points file in the I/O application programs developed using SCOLL, a macro language provided by SURFACER. The points file is read into a database in the model application. The model application is a group of programs developed using VB, it is used to segment the points and access to database by calling ODBC. When the geometric model is created, a model file can be established in the model application. The model file can be read in SURFACER for displaying by the I/O application. All above software tools and programs will be integrated under the Window environment in a PC.

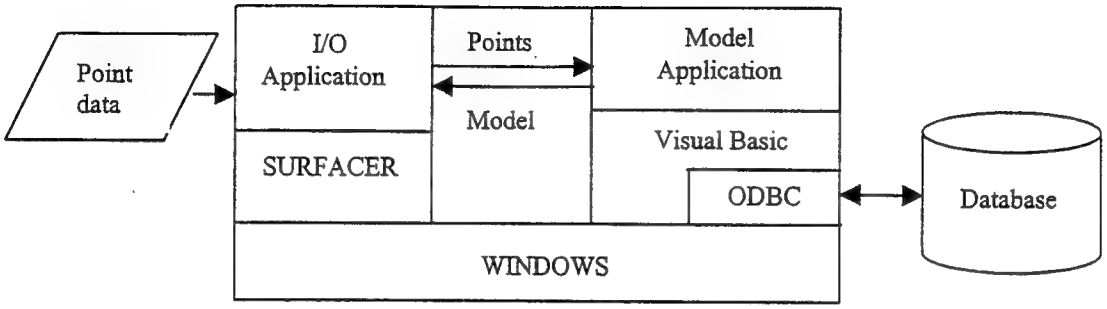


Figure 14. The development environment

4.2 System Architecture

Based on the project features, the tasks can be divided into segmentation and fitting-oriented application, interface to ODBC API and access to point data source. The application and interface would be programmed with VB, and the access to data source is then realized by database engine itself. They are constructed together in a level structure. As shown in Figure 15. where application frame includes a group of main programs, which are used to control the execution of the programs, calculate some parameters, process unusual situation, call the subroutines for segmentation and fitting, and so on. The programs interfacing to ODBC and the application frame are then grouped together in the module of subroutines interfacing to ODBC. They are computer programs that call ODBC API according to the requests sent by application frame to create a database, extract related points from database with some constrains, and import the result calculated by application frame into database, etc. In addition, the extracted data is also returned to application frame through them. As for the ODBC manager/server, it is a DLLs library provided by Microsoft, and is used to access to data source directly. All information data will be stored in database as source data. These source data include the point data, surface data, and temporary results produced during the execution procedure of the system.

4.3 Database Structure

In RDB (relational database), the database structure consists of three levels, namely page level, table level, and record level. The entities of an object can be represented by a group of tables. The entity's content or attributes are

described with the fields in a record. The table's information, such as table structure, table size, and the relationship between tables will be recorded in the page level.

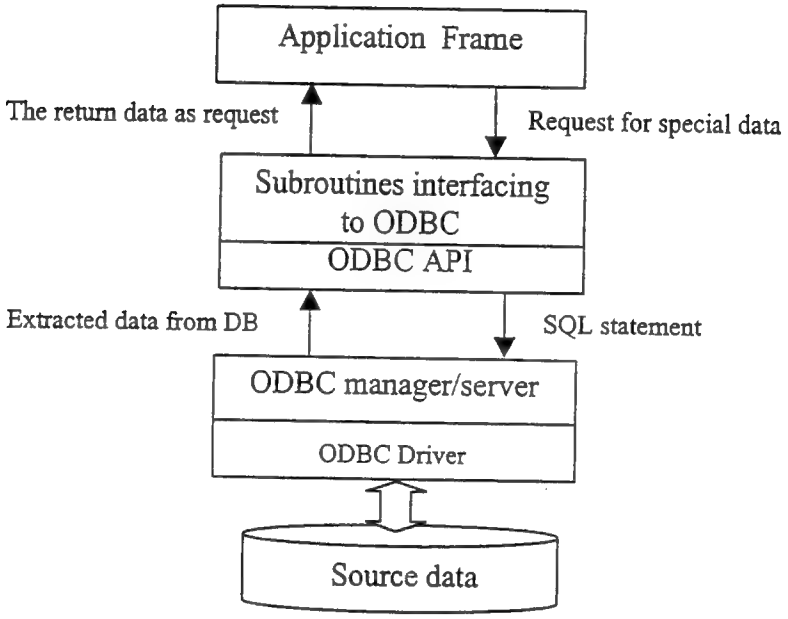


Figure 15. System architecture

The data structure of a relation database is simple. It simply uses 2D tables to describe an object and the relationship between the entities of an object. Generally, some complex relationship between entities of an object can be represented effectively with RDB tables.

4.3.1 Defining and Creating Tables

4.3.1.1 Defining Tables

In this project, the input is a group of point coordinates, and the output is a group of surfaces, which are segmented out by the programs. During the execution of the programs, some temporary results will be produced, such as the normal vector at each point, the marks which identify whether the point has been processed, which face the point

belongs to, and whether it is a boundary point, etc. According to these data, we defined three tables to describe these data information. They are discussed respectively as below.

The point data is first constructed in the original point table, Point for short, as shown in Table 2. This table will be referencew when precise point data is needed.

Table 2. Original point table

Index	X	Y	Z

Where, Index: a string value, which takes the value of: str(I)+","+str(J)+","+str(K);

X: x coordinate value, Double;

Y: y coordinate value, Double;

Z: z coordinate value, Double.

To conveniently locate the neighboring points around a point, a table called cell-occupied matrix table, COM table for short, is defined, as shown below. The method of obtaining the matrix can be found in Section 3.

Table 3. Cell-occupied matrix table

I	J	K	X	Y	Z	Vx	Vy	Vz	Em	Sm

Where, I, J, K is the sequence number of a cell which include a point in the cells matrix along X,Y,Z axis direction. They all are key index;

(X, Y, Z) is an average point coordinate of the points that allocate in a same cell, so that each cell contains only one point;

(Vx,Vy,Vz) is the average normal vector in the point;

Em is a mark identifying whether a point is on boundary, it takes an integer value, 0 means not on, 1 means on;

Sm is also a mark identifying which surface a point lies on. It is string value, and takes the value of ID that is defined in the surface table (see following)

The surfaces segmented will be stored in a table named surface table. It mainly records the surface's identification, type and parameters. The segmented points on the surface will be marked in the cell-occupied matrix table.

Table 4. Surface Table

ID	Type	P1	P2	P3	P4	P5	P6	P7

Where ID: is the identification of a surface, it assumes an integer value;

Type: means the type of a surface, including plane, cylinder, and sphere;

P1-P7 are the parameters for a surface. When the surface is a plane, P1-P4 will be A,B,C,D of the equation A*X + B*Y + C*Z + D = 0;When the surface is cylinder, (P1,P2,P3) indicate the original point (X,Y,Z) of cylinder axis, and (P4,P5,P6) represent the normal vector (l,m,n) of the axis. P7 is the radius value of a cylinder; When the surface is a sphere, then (P1,P2,P3) indicate the center of sphere, P7 is the radius of sphere.

4.3.1.2 Creating Tables in VB

Each database in a Visual Basic database has a collection of table definitions. The collection object is the tabledefs collection, and each individual member of the collection is a TableDef object. Each TableDef object, in turn, has a collection of field. The collection object is the Fields collection, and each individual member of the collection is a Field object, each filed object has a set of properties. A new table can be created in a database by a three-step process:

- Create a TableDef object by assigning the return value of the Database object's CreateDatabaseMethod to a TableDef object variable. The variable db is a valid database object, it should be defined in a program.

- Create a Field object by assigning the return value of the TableDef object's CreateField method to a Field object variable. The field's name property and type property at the time it is created must be provided. After each field is created, It can be added to the field to table using the Append method of the TableDef object's Fields collection.

- When all fields of a table have been created, the TableDef object can be added to the database with the append method of database object's TableDefs collection.

4.3.2 Indices and The Relations between Tables

In order to speed the searching and make searching convenient from the three tables, the relations between these tables should be created. The relations between tables in the RDB are represented by indices. Indices can be broadly classified into two types: primary key and other indices.

A relation depends on a key field, a field that the two tables in the relation have in common. In the base table, the table on the "one" side of the relation, the key field is called the primary key. Every table may have one, and only one, primary key. If a table is to be used as the base table in a relation, it must have a primary key. The primary key can be single field. Or it can be built from multiple fields. A key built from multiple fields is known as composite key. The primary key can serve as a unique identifier of records in a table.

Additionally, getting the best performance from a database requires that indices are established on appropriate fields. Indices have three principal benefits:

- Allow to designate the order in which records from a table -type recordset are presented to the user;
- When a field is defined and table-type recordset is being used, records matching the indexes values can be rapidly found through the recordset objects's seek method;
- When an index field is used in the Where or Order By clause of an SQL select statement, the engine can use it to optimize queries.

In our project, there is very huge number of data, and it is always needed to search and choose out a group of record set, consequently, the above three benefits are very suitable to the project requirements.

For the COM table, we often need to choose a group of points that have a special I, J or K value, so that the fields I, J, and K would be defined as indices. When a surface has been segmented out, the points that lies on the surface should be able to be extracted out from the COM table. So that a relation between the SURF table and COM table should be created. The relation is 1:n type for SURF to COM. SURF's field ID is defined as a primary key, and COM's field Sm is defined as a foreign key. In addition, based on the recordset extracted by the ID primary key, the boundary of each surface can also be derived using the Em field values. As to the points table, the index field is defined an index, it takes the combination of (I,J,K) as its value. Through it, the points contained in a same cell can be extracted.

4.3.3 Interface to Database

Database engine includes a lot of operators on data source, such as Creation, Define, Update, Selection, Projection, Join and Union, etc. Through these operators, a user can establish the database structure, store a new data into the database, update the related values in a table, and obtain the related data he wants. In VB, the database engine to access database is provided based on the ODBC standard, so that a series of SQL operators can be used to access the source data.

When the programs of segmenting the point clouds executed, the following operations on database are often called:

- extract a group of points which satisfy some constrains;

- mark a field for recording the status;
- input some new information into database;

All of the operations can be realized by calling the database engine. For example, when we want to extract all cells which have the same I=m value, the calling subroutine using SQL statement looks like below:

```
Select COM.[I]
From COM
Where COM.I = m
Order By [ I ]
```

In Visual Basic, the program is:

```
Dim rs As Recordset
Dim sql As String
sql = "Select COM.[I]" & "From COM" & "Where COM.[I] = m" & "Order By [I]"
rs = db.OpenRecordset (sql, dbOpenSnapShot)
```

From the example, it can be concluded that VB and ODBC is suitable to be used as the development language in this project and the database structure defined as above is also applicable.

5. Results and Discussion

In this section, the entire procedure of automatically creating a CAD model will be described. This procedure includes: import original point data, segment the point data, group points by features, and create the wire-frame and surface models. Original data will be described by their property, type and format Also the method where original data come from will be introduced and then the table where the data and the result will be saved in will be shown. Next, the most important factor, cell size, will be discussed using an example. Finally, the advantage and disadvantage of the proposed methodology will be discussed.

5.1 Original Data

To test the accuracy of the developed algorithm, a test part is created from the Surfacer software using sample method. After creating the point cloud, excessive points were deleted by manual ways, for example overlapping surface between two features. The original whole point model and the original whole surface model are shown in Figures16 and 17, respectively. The inner features of the test part are also displayed in Figure 18.

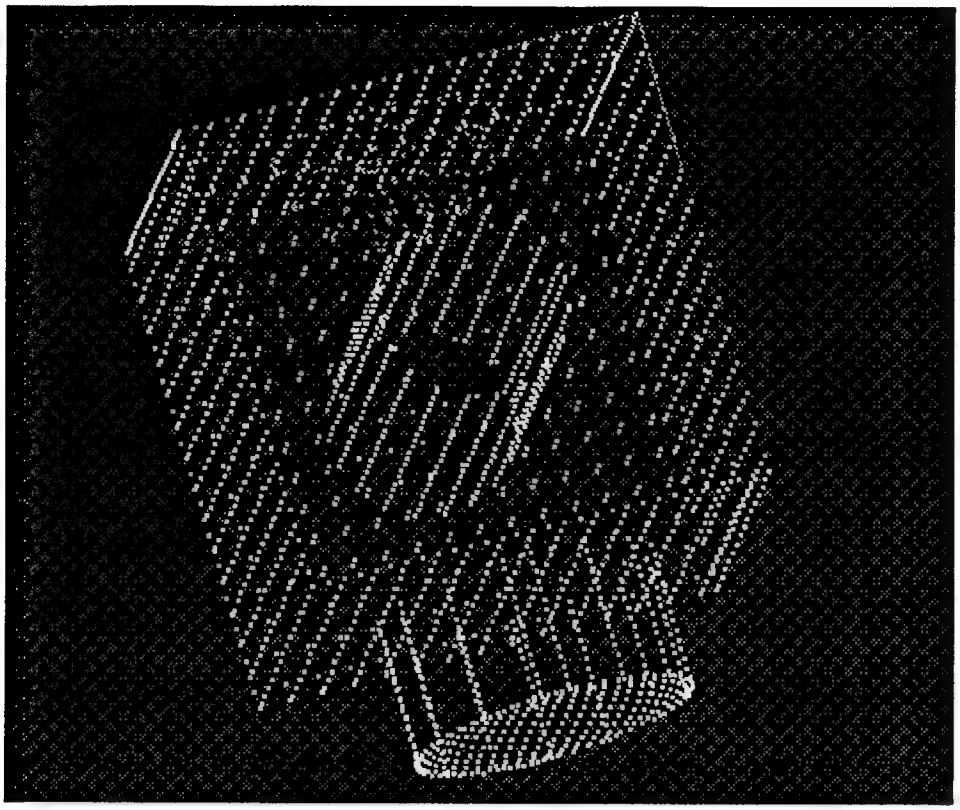


Figure 16. Original point model

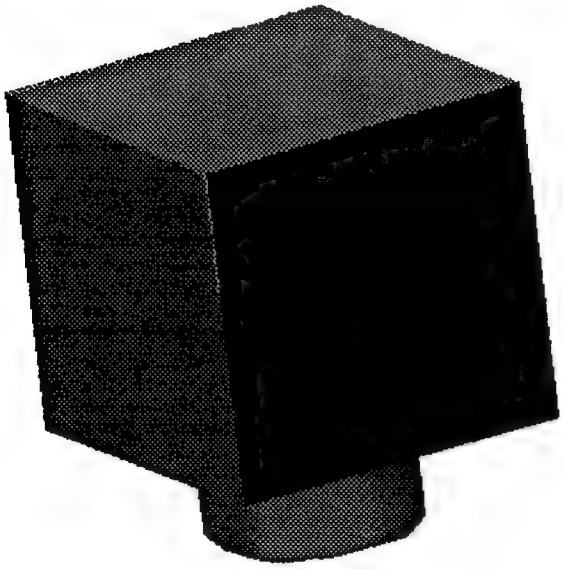


Figure 17. Original entire surface model

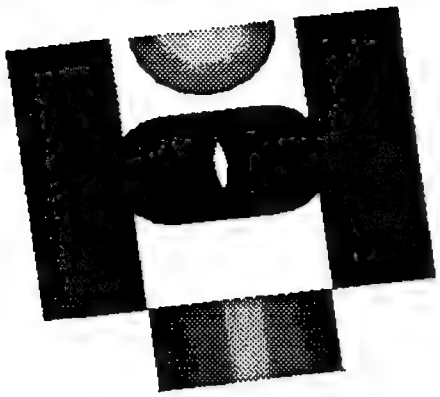


Figure 18. Original inner feature model

5.1.2 Cell Size

After importing data, the system needs a cell size for importing IJK table, as discussed in Section 4. The most important factor whether this model processing can be successfully accomplished depends on the accuracy of the cell size. Cell size should be calculated based on point density, size of the part, and precision of scanning. But the cell size used in this project comes from empirical data. In practice, cell size plays such an instrumental role that in extracting geometric characteristic method, it can be used to differentiate whether the point belong to a line or a surface. And the empirical data is also deducted from observing the points. In this research, the cell size is set as 0.01", because this value is large enough to ensure to have at least one point in each cell in a such area which points are available.

5.1.3 Postprocessing

The Surfacer software is a CAD software, it can create standard surface model from point model and find the surface boundaries. It can get data from text file and save them as IGES standard files. Using this software, the developing time will be greatly decreased. This software provides a API for Visual C++, but for the time and funding limitations, the API program was not bought. If this program was used in this research, the developing time will even be shorter than what it took at present. In this research, some commands from the Surfacer software such as surf_edge_to_curve3, fit_plane, fit_cylinder, fit_sphere, and make_isopar_curve3, were applied.

5.2. Results and Discussion for the Test Part

The results obtained from the test part are shown in Figure 19, Figure 20, and Figure 21. These research results are considered as excellent. There are several reasons for this.

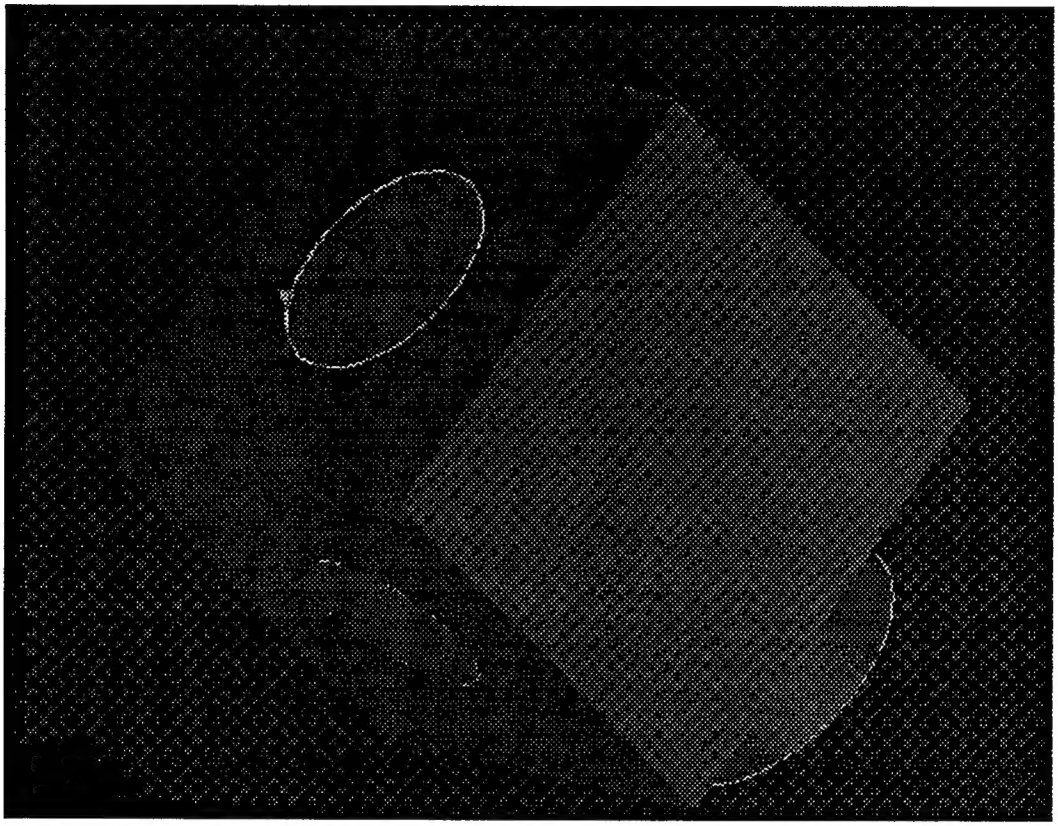


Figure 19. Entire Surface Model

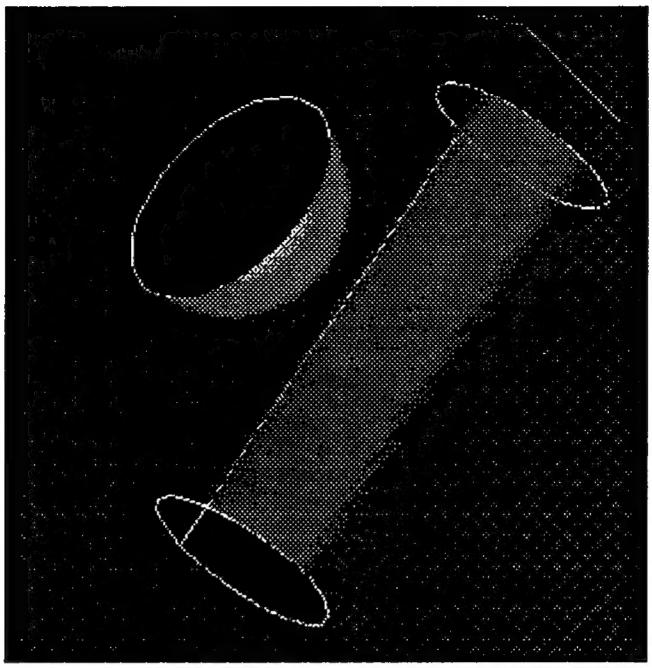


Figure 20. Inner Surface Model

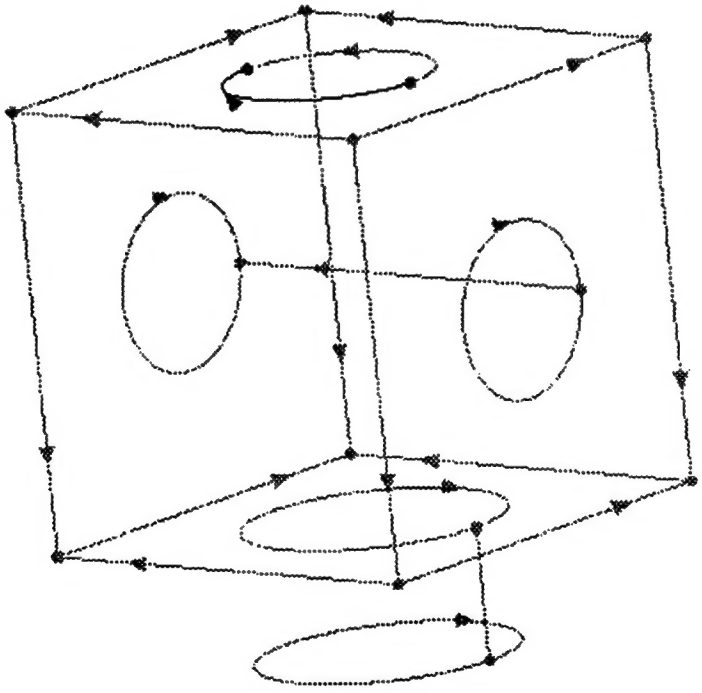


Figure 21. Entire Wire-Frame Model

First, no any real surface model function was obtained from the modeling system, actually, this modeling system classify the points lying on same surface into same group. The real information of surface model was got from Surfacer software. Because Surfacer has mature technology, the information of surface is accurate and all results

saved in IGES standard files. So it helps to get precise result. Secondly, the original data is accurate and same applies to its vector value, so the final result deducted from it is also accurate. Third, all features parallel to XYZ axis and no noise or free-form existed, so no misjudge will happen.

But there are no way to guarantee the original points are accurate, all of the original points need pre-process. Surfacer provide some method to pre-process points, such as reduce points , add points. In some cases, reduce the point accuracy is one way can be relied on. For example, if the original accuracy is 10^{-6}, by reducing its accuracy to 10^{-3}, the range of tolerance is enlarged . Though the tolerance is reduced, a result can be deducted finally. Another way is to select a reasonable cell size. Cause there are negative and positive tolerance, one result can be obtained from this research that the tendency of tolerance has its regularity. The average value among one cell tends to be accurate if there are enough points in this cell. So it is very important that the cell size is selected precisely.

In this research, there are approximately 6,000 points. The part size is 15cm in each axis and it has 10 surfaces, seven of them are cylindrical surfaces, two of them are plane surfaces, one of them is sphere surface. The part has seventeen edges, five of them are arch, twelve of them are lines. Testing process from importing point data to getting wire-frame model cost 30 minutes. Several factors affect run time. First, this project codes in VB. In scientific calculation, VB is not the most efficient one, its running speed is lower than C^{++}, and there are no API between Surfacer and VB. So many method in Surfacer can not be used in this program. Moreover, database in Surfacer cannot be used in VB, and ODBC in VB runs slowly. Finally, the hardware is another reason affect running time. In this project, only Pentium 200, 64 MB memory is available, and program needs to be read and write data frequently from hard drive. Such process consume more running time. If hard ware improves, the total running time will reduce in a great deal.

6. Conclusion

In this research, a method was developed that allows a complete point cloud to be automatically segmented and thus fitted with curves and surfaces to create a geometric model for the parts consisted of standard surfaces, such as plane, cylinder, sphere, cone and so on. By extracting particular geometric characteristics, the type of surface can be determined and the corresponding surface parameters can be calculated. The points on this surface can then be grouped, segmented out, and fitted with a standard surface, with its boundaries fitted with curves. This research provides advantages over current methods of edge-based and face-based segmentation by avoiding an iterative process and directly extracting the surface features that are essential for subsequent manufacturing procedures, such as CAPP and CAM.

This research deals only with the standard surfaces of the *Plane*, *Cylinder*, *Sphere*, *Torus*, and *Cone*, and thus only these types of surfaces can be automatically segmented and fitted. Nonetheless, most of the features found in

industrial products are composed of these types of standard surfaces, and so this research already provides an effective and practical methodology for segmentation.

Freeform surfaces can be detected using the method in this research, but they still must be segmented and fitted interactively using a reverse engineering software program. In order to completely automate geometric model creation for all surface types and increase the scope to include all types of products, freeform surfaces must be investigated in future research. The research performed here can nevertheless be used as a foundation for future work in the continuous endeavor to accomplish true complete automation of digitization-based geometric model creation.

For testifying the availability of the methodologies proposed, a system was implemented that can be used to create a geometric model automatically based on point data. The system was build using Visual Basic and ODBC, and tested with a sample part, which were created in SURFACER by sampling the surface model. The results have declared that the methodologies are applicable, and the effectiveness is also satisfactory.

7. References

1. Smith. DR and Kanade, T "Autonomous scene description with range imagery", CVGIP 31(3), 1985, pp 322-334.

2. C.Bradley, G. W. Vickers, and M. Milory, "Reverse engineering of quadric surfaces employing three-dimensional laser scanning", Proc. Inst. Mech. Eng. 208, 1994, pp 21-28.

3. Leonardis, A. Gupta, A and Bajcsy, R "Segmentation of range images as the search for geometric parametric models", International Journal of Computer Vision 14,1995, pp 663-675.